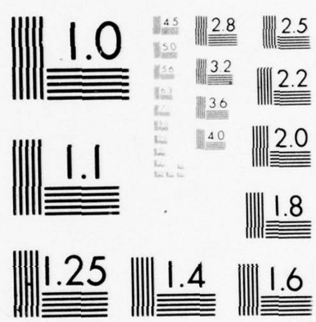
AD-A052 150 CALIFORNIA UNIV LOS ANGELES DEPT OF SYSTEM SCIENCE F/G 17/2
ALGORITHMS FOR THREE AND FOUR DIMENSIONAL GEOMETRICAL MOMENT SP--ETC(U)
FEB 78 M A KING, K YAO N00014-76-C-0875
UNCLASSIFIED UCLA-ENG-7806 NL

| OF |
AD
A052150

END
DATE
FILMED
5-78
DDC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

UCLA-ENG-7806
FEBRUARY 1978

12

Prepared for the
Office of Naval Research
under Contract N00014-76-C-0875

Principal Investigator – K. YAO

ALGORITHMS FOR THREE AND FOUR DIMENSIONAL GEOMETRICAL MOMENT SPACE BOUNDING

M.A. KING, JR.

UCLA • SCHOOL OF ENGINEERING AND APPLIED SCIENCE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)

6 Algorithms for Three and Four Dimensional Geometrical Moment Space Bounding.

5. TYPE OF REPORT & PERIOD COVERED

9 Technical Report, 1978

6. PERFORMING ORG. REPORT NUMBER

14 UCLA-ENG-7806

7. AUTHOR(s)

10 M. A. King, Jr. K. /Y30

8. CONTRACT OR GRANT NUMBER(s)

15 N00014-76-C-0875

9. PERFORMING ORGANIZATION NAME AND ADDRESS

Department of System Science
School of Engineering and Applied Science
University of California, Los Angeles, CA 90024

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS

Office of Naval Research
Electronics Program, Code 427
Arlington, VA 22217

12. REPORT DATE

11 Feb[redacted] 1978

13. NUMBER OF PAGES

84 12 79P.

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)

15. SECURITY CLASS. (of this report)

Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release: Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Digital Communications -- Error Probability -- Moment Space Bound -- FORTRAN IV Program -- Intersymbol Interference.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The solution to many problems in communication theory takes the form of a moment of a function of a random variable. Often this moment is difficult to evaluate numerically. When this is the case, tight bounds to the true value are sought that are relatively easy to evaluate. One method of deriving such bounds is a geometrical technique that is a result of an Isomorphism Theorem from Game Theory. Recently, very useful bounds have been derived with this technique using two and some classes

next page

405 213 JOB

of three-dimensional geometries. All of these results have been analytical in nature. This report extends this work by providing algorithmic techniques for evaluating bounds produced by all classes of three and four-dimensional geometries. In addition, a procedure for extending these algorithms to problems of dimensionality greater than four is outlined. Implementations of the three and four-dimensional algorithms are presented in appendicies.

TABLE OF CONTENTS

I. INTRODUCTION

1.1 Problem Description

There are many important problems in the field of communications theory that have as their solution the expectation of a random variable. Perhaps the classic example of such a problem is that of computing the probability of bit error for a binary signal being transmitted on a channel with linear intersymbol interference [1] - [15]. A block diagram for this example is given in Figure 1.1.

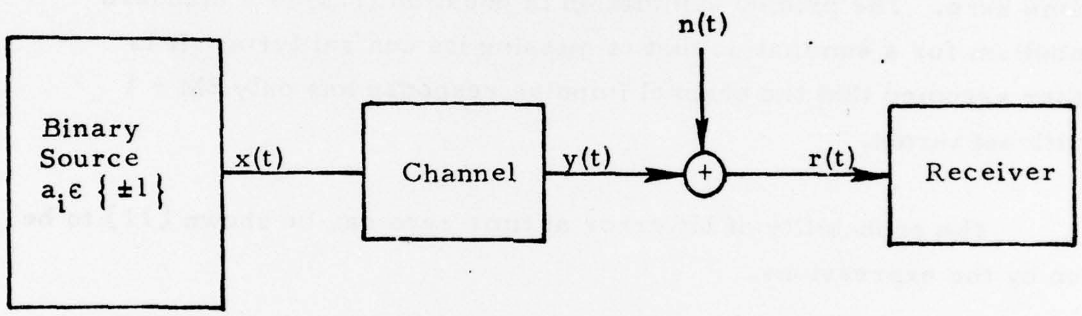


Figure 1.1

The binary source selects a value for a_i with equal probability each T seconds. These source symbols are encoded into waveforms suitable for transmission across the channel. The time function x(t) represents a string of these channel waveforms. The channel is assumed to act upon the waveform string x(t) as a linear filter. Thus, the waveform associated with a particular source symbol will typically be distorted in shape and spread in time by the action of the channel. Let the distorted waveform string at the channel output be represented by y(t). This signal is assumed to be further distorted by the addition of a white Gaussian noise process, that is denoted by n(t). The waveform string finally presented to the receiver is r(t) where

$$r(t) = y(t) + n(t). \tag{1.1}$$

This signal is detected and sampled. The sampled output at time zero can be represented by

$$r_o = a_o h_o + \sum_{i=-M}^{M}{}' a_i h_i + n_o \tag{1.2}$$

where r_i is the detected and sampled output at time i, ... a_{-M} ... a_{-1} a_o a_1 ... a_M ... is the binary input signal string, $\{h_i\}$ is the sampled impulse response of the channel, and n_o is the Gaussian noise sample at time zero. The primed summation in equation (1.2) is a standard symbolism for a summation that is missing its central term. It is further assumed that the channel impulse response has only 2M + 1 significant terms.

The probability of bit error at time zero can be shown [11] to be given by the expressions

$$P_e = E_U \left[Q\left(\frac{h_o + u}{\sigma}\right) \right] \tag{1.3a}$$

$$= E_U \left[\left(Q\left(\frac{h_o + |u|}{\sigma}\right) + Q\left(\frac{h_o - |u|}{\sigma}\right) \right) /2 \right] \tag{1.3b}$$

where

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^{\infty} \exp(-y^2/2)\, dy, \tag{1.4}$$

$$u = \sum_{i=-M}^{M}{}' a_i h_i, \tag{1.5}$$

σ is the standard deviation of the Gaussian noise, and U represents the space of all strings of 2M binary symbols. The expressions (1.3a) and (1.3b) are clearly equal mathematically, but it has been shown [11] that one form or the other can have analytical advantages when evaluating the probability.

Unfortunately, the expressions in equations (1.3a) and (1.3b) may be difficult or computationaly impractical to solve exactly. For example, if the interymbol interference extends for forty samples preceeding and trailing the actual signal sample time (M=40), the exact evaluation of the probability of error would involve the summation of $2^{80} \approx 10^{24}$ terms of the form of equation (1.4). Thus, even if equation (1.4) could be solved in 1 nanosecond of computer time, exact computation of P_e would require 3×10^5 centuries. Although channels having an impulse response that is significant over 80 bit times may be rare, it is clear that the computation involved in calculating (1.3a) or (1.3b) can still be large even for fairly modest impulse responses.

Expressions similar to (1.3a) can be derived for the probability of bit error on any additive Gaussian noise channel with linear interference. Examples would include spread spectrum multiple access channels, and channels with co-channel interference [16] - [19]. Thus, the evaluation methods that will be discussed below are more generally applicable than to just intersymbol interference problems. They will apply to Gaussian channels with other kinds of linear interference as well.

1.2 Isomorphism Theorem

One possible approach to problems in communications and information theory that appear difficult or impossible to solve in their exact form is to find bounds to the exact solution that are easily computed. One technique that has been proven to be useful in providing bounds to problems of this kind is the Moment Bounding Technique [11] - [19]. This technique is based on an Isomorphism Theorem from Game Theory [20], [21]. In this approach, the moment of the function of the random variable that is of interest is bounded in terms of moments of other functions of the same random variable. These other functions are chosen such that their moments are relatively easy to evaluate. This approach has the unique advantage that both upper and lower bounds can be found with the same computational technique. In addition, the moment bounds that are derived using two or three dimensions yield a relatively simple geometrical understanding of the bounding process.

The Isomorphism Theorem can be stated as follows:

Isomorphism Theorem:

Let u be a random variable with probability distribution $G_U(u)$ defined over a finite closed interval $I = [a, b]$. Let $k_1(u), k_2(u), \ldots, k_n(u)$ be n continuous functions defined on I. Let m_i, $i = 1, \cdot \cdot \cdot, n$, denote the n generalized moments of the random variable u induced by the functions $\{k_i(u)\}$.

$$m_i = \int_I k_i(u)\, d\, G_U(u) = E_U[k_i(u)], \qquad i = 1, \ldots, n \tag{1.6}$$

Denote the moment space $\mathcal{M}$ as

$$\mathcal{M} = \{ \underline{m} = (m_1, m_2, \ldots, m_n) \in R^n \} \tag{1.7}$$

where $G_U(u)$ ranges over the set of all probability distribution functions defined on I. $\mathcal{M}$ is a closed, bounded, and convex set.

Let $\mathcal{C}$ denote the generalized curve $\underline{r} = (r_1, r_2, \ldots, r_n)$ traced out in R^n by $r_i = k_i(u)$ for $u \in I$. Let $\mathcal{H}$ be the convex hull of $\mathcal{C}$. Then $\mathcal{M} = \mathcal{H}$.

The application of the Isomorphism Theorem to bounding problems can be seen from the following two dimensional ($n = 2$) example. Given a function $k_1(u)$ of the random variable u whose moment, $E_U[k_1(u)]$, is desired, select a second function, $k_2(u)$, whose moment is easily computable. By identifying the functions $k_1(u)$ and $k_2(u)$ with the two orthogonal axes of a two-dimensional coordinate system, as in Figure 1.2, a curve $\mathcal{C}$ can be traced out as u varies through its finite range of values. The convex hull, $\mathcal{H}$, of the curve $\mathcal{C}$ can now be found, as in Figure 1.3. Let m_2 denote the value of the moment of the function $k_2(u)$ as in (1.6). According to the Isomorphism Theorem, the set of all moment pairs

$$\mathcal{M} = \{ (m_1, m_2) \mid m_1 = E[k_1(u)]\ ,\ m_2 = E[k_2(u)] \} \tag{1.8}$$

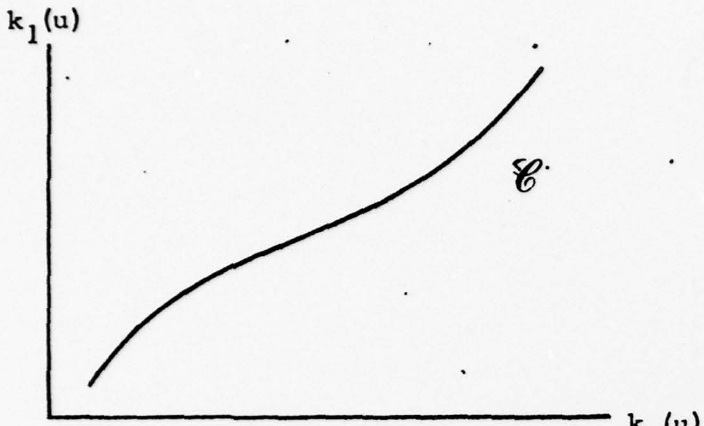


Figure 1.2

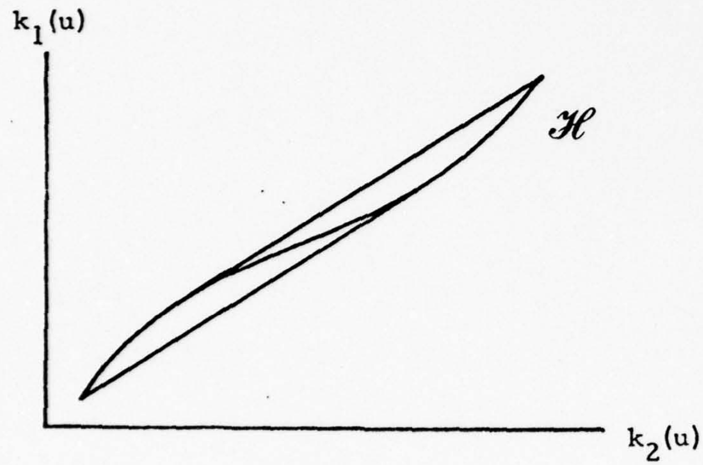


Figure 1.3

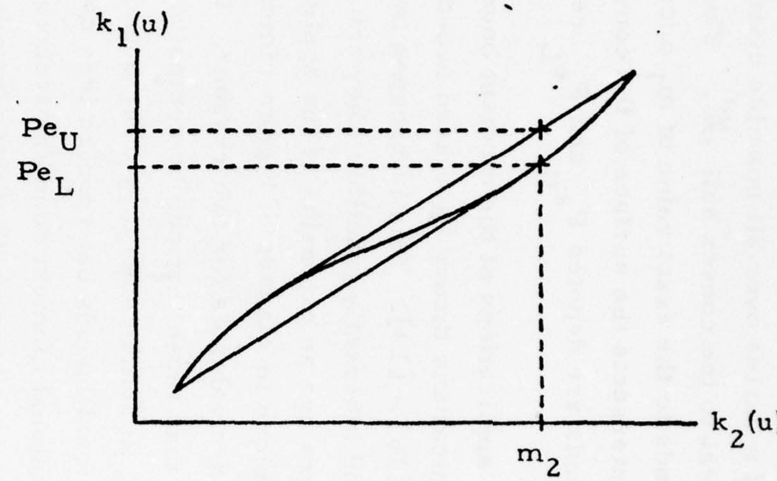


Figure 1.4

as the distribution of u varies over all possible distributions defined on the range of u, is identical to the convex hull $\mathcal{H}$. Thus, from Figure 1.4, upper and lower bounds to the exact value of m_1 occur at the points where the line $k_2(u) = m_2$ intersects the surface of the convex hull. In Figure 1.4, the values of the bounds are denoted P_{e_U} and P_{e_L} respectively.

Most of the applications of the moment bounding techniques to problems in communications theory have used two-dimensional moment bounds [8] - [13], [16] - [19]. This is because two dimensional bounds are quite intuitive and inherently tractible (they can always be found graphically). The following chapters are an extension of the applications of the moment space bounding technique to classes of higher-dimensional bounds. Higher dimensional bounds are valuable for two reasons. First, they usually offer tighter bounds than those that can be computed with two-dimensional techniques. Second, in some cases they offer bounds that are as tight as the best two-dimensional bounds but require less computational effort. Thus a higher-dimensional moment bounding technique offers tigher bounds, or less effort, or in some cases both tighter bounds and less effort than two-dimensional bounds.

II. THREE-DIMENSIONAL MOMENT BOUNDING ALGORITHM

2.1 Introduction

Most of the research effort that has been expended on applications of the moment bounding technique [11] - [19] has been expended deriving exact analytic expressions for the upper and lower bounds. The forms of the expressions for the bounds are typically conditional on the parameters of the curve $\mathscr{C}$, and on the values of the auxillary moments. For example, consider the two-dimensional intersymbol interference moment bound results reported by Yao and Tobin [11]. In particular, consider the case of the exponential auxillary bounding function. The equations for the curve $\mathscr{C}$ for this example are

$$x = k_1(u) = e^{c(h_o + u)}, \tag{2.1}$$

and

$$y = k_2(u) = \operatorname{erfc}\left(\frac{h_o + u}{\sigma}\right) . \tag{2.2}$$

In these equations, c is an arbitrary constant, erfc (·) is the standard complementary error function, u is the amount of intersymbol interference, h_o is the amplitude of the desired signal, and σ is the standard deviation of the additive white Gaussian noise. As is demonstrated in [11], for given values of h_o and σ and given limits on the range of u, the curve $\mathscr{C}$ will vary from being convex $\cap$, to "S-shaped", to convex $\cup$, depending on the value of the parameter c in (2.1). Clearly the form of the analytic expressions for the boundry of the convex hull generated by this curve is a function of the parameter c. In addition, when the curve $\mathscr{C}$ is "S-shaped", the form of the bounding expressions will depend on the location of the bounds on the surface of the convex hull. This is to say, they will depend on the value of the auxillary moment $m_1 = E_U[k_1(u)]$ of equation (2.1). These facts are illustrated in Figure 2.1.

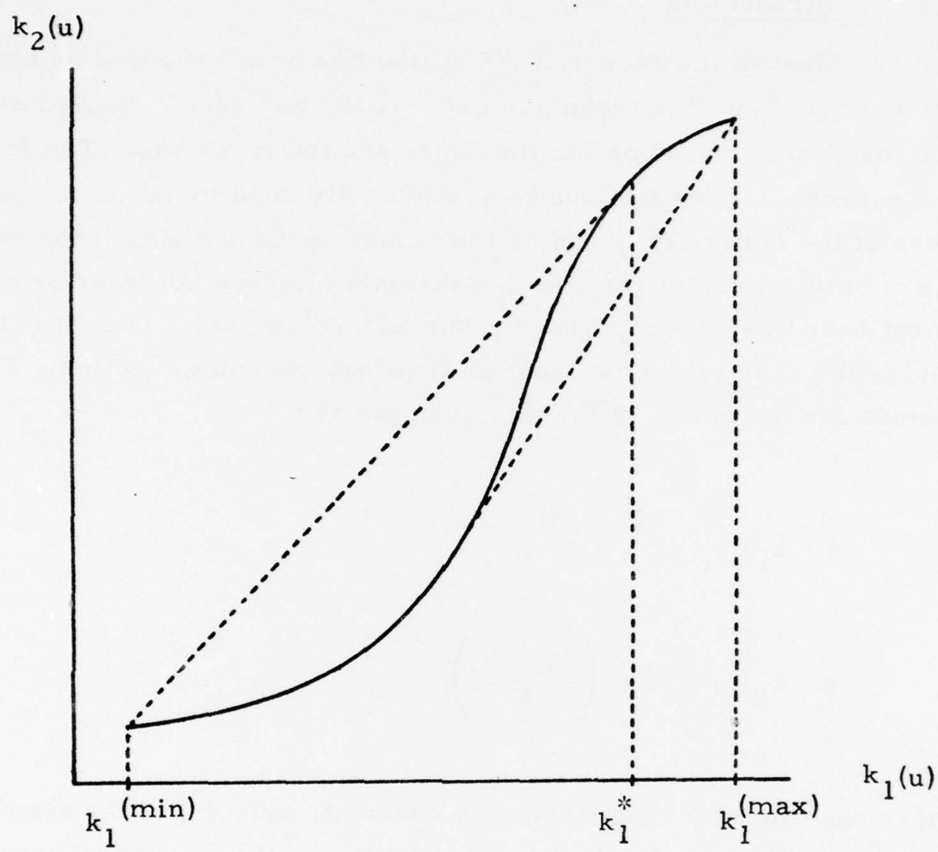


Figure 2.1

It is seen in Figure 2.1 that the form of the upper bound will be linear in the region $(k_1^{(min)}, k_1^*)$ of the abscissa, but will take on the form of the curve $\mathscr{C}$ in the region $(k_1^*, k_1^{(max)})$. Thus, the equations for the bounds derived using the moment bounding technique are not typically of a single form, but are composites, strongly dependent on the parameters of the bound.

From the standpoint of a casual user of the bounding technique, the composite nature of the bounding equations is inconvenient. It would be more convenient if some sort of unified computational approach were found. Such an approach would eliminate the need to make detailed preliminary investigations into the local geometry of the curve $\mathscr{C}$ and its convex hull $\mathscr{H}$. The bounding technique could then be used more easily on a broad spectrum of bounding problems.

An algorithmic approach is such a unified approach. In such an approach, the curve $\mathscr{C}$ is represented in a computer as an array of points. This array is a set of samples taken in some systematic manner along the length of $\mathscr{C}$. The algorithm would then construct the convex hull of this set of samples. The values of the bounds could then be computed by operating on this modified hull.

The algorithmic approach derives its unifying properties from the fact that it deals with an array of sample points taken from the curve, *rather than dealing with the smooth curve itself.* This is equivalent to replacing the curve $\mathscr{C}$ with a piecewise linear approximation to $\mathscr{C}$. In three-dimensions the convex hull generated by this approximation will always be a polyhedron. The hull generated by the original curve could have been any three-dimensional convex figure. Evaluating the bounds then reduces to finding the appropriate planar faces of the polyhedron, and then computing the coordinate values of the appropriate points on these faces. This general procedure is independent of both the shape of the curve $\mathscr{C}$ and of the local properties of the original convex hull $\mathscr{H}$, generated by $\mathscr{C}$.

In the following sections an algorithm is presented that will compute the upper and lower moment bounds for the three-dimensional moment bounding problem. It will be shown that the algorithm is reasonably modest in its use of central processor (cpu) time for a representative example. Both a general description of the procedure and a listing of an actual implementation will be presented. The technique is based on the work of Appel and Will [26].

It is noted that this procedure computes an approximation to the moment bounds. There are two responses to this observation. First, since the curve $\mathscr{C}$ is assumed to be smooth and of finite length, the sample points in the array can be chosen to be sufficiently dense to assure that the results will approximate the true value of the bounds to any required accuracy. Secondly, the algorithmic solutions can be adjusted to lie outside the original bounds. A method of computing the adjustments will be presented below. Thus, the adjusted approximations will always be valid bounds. With the adjustments they will not be as tight as the exactly computed moment bounds, but the adjustments can be made arbitrarily small by making the sample points sufficiently dense.

The arguments concerning the adjustments to the algorithmic solutions are as follows. By the definition of the convex hull of a curve, the hull generated by any piecewise linear approximation to the curve $\mathscr{C}$ will be interior to the hull generated by $\mathscr{C}$ itself. Thus, as the number of sample points of $\mathscr{C}$ increases, the hull of the approximation will approach the original hull, $\mathscr{H}$ from the inside.

Denote the piecewise linear approximation to the curve $\mathscr{C}$ by $\hat{\mathscr{C}}$. Let $\delta(p)$ be the minimum Euclidean distance from point p on the curve $\mathscr{C}$ to $\hat{\mathscr{C}}$. Let δ be the largest such distance for any p on $\mathscr{C}$. Denote the convex hull generated by $\hat{\mathscr{C}}$ as $\hat{\mathscr{H}}$. Consider the polyhedron that is **similar** to $\hat{\mathscr{H}}$ but at a minimum Euclidean distance of δ to the outside of $\hat{\mathscr{H}}$. That is, the polyhedron that has planar faces parallel to the planar faces of $\hat{\mathscr{H}}$. Clearly, the original curve $\mathscr{C}$ must be interior to this new polyhedron. Similarly, the original convex hull $\mathscr{H}$ must be interior to this new figure. Therefore, if the upper bound computed by the algorithm is increased by an amount δ, and the lower bound decreased by δ, the resulting numbers are guaranteed to be true bounds to the desired moment.

2.2 Preliminary Definitions

Let A denote a three-dimensional array containing a finite number of elements. These elements may be thought of as being the coordinate values of sample points of a smooth twisted curve $\mathscr{C}$ in E^3. Let $\hat{\mathscr{H}}$ denote the three-dimensional convex hull generated by A. Let F denote a plane and $\mathscr{P}_F$ denote the usual orthogonal projection operation from E^3 onto F. Then there is an array A_F on the plane F such that

$$A_F = \mathscr{P}_F A. \tag{2.3}$$

The array A_F is the projection of the array A onto the plane F.

Let the convex hull of the set A_F be denoted by $\hat{\mathscr{H}}_F$. A basic property [22] of convex bodies can be stated as

$$\hat{\mathscr{H}}_F = \mathscr{P}_F \hat{\mathscr{H}} \tag{2.4}$$

This is to say, the convex hull of a projection (A_F is the projection of A) is equal to the projection of the convex hull. This property is fundamental to several different algorithms (e.g., [26], [27]) that compute three-dimensional convex hulls. It will also be basic to the algorithm developed below.

The property that is expressed as equation (2.4) will be used to reduce the problem of determining the structure of a three-dimensional convex hull into a sequence of two-dimensional convex hull problems. There are satisfactory computational methods known for finding the convex hulls of two-dimensional arrays [26] - [29]. A modified version of one of these methods [26] will be used in the development of the algorithm that follows.

The convex hull of a finite element two-dimensional array A_F will be a convex polygon, $\hat{\mathscr{H}}_F$. The sides of this polygon will be chords connecting members of the set A_F. These chords are shown by equation (2.4) to be the F-plane projections of chords joining points of the array A that are on the surface of $\hat{\mathscr{H}}$ in E^3. Thus, if there are a finite number of elements in the array A, there must be a finite length sequence of projection planes $\left| F_n \right|$ that will allow the identification of the entire network of chords that lie on the surface of $\hat{\mathscr{H}}$. This network can be seen to define the planar faces of the polyhedral convex hull $\hat{\mathscr{H}}$. Therefore, the network effectively defines the entire surface of $\hat{\mathscr{H}}$.

One method of constructing a sequence of projection planes $\left| F_n \right|$ involves the "collapsing" of surface chords of $\hat{\mathscr{H}}$. This method is developed in [26], and is the method that is used in the algorithm presented here. In this method, an arbitrary first projection plane F_1 is selected and the correspoinding projected hull $\hat{\mathscr{H}}_{F_1}$ is determined. A boundry chord of $\hat{\mathscr{H}}_{F_1}$ is selected in some systematic manner. Denote the selected chord by Ch_{F_1}. The chord Ch_{F_1} is the F_1-plane projection of a chord (to be denoted Ch_1) that is on the surface of $\hat{\mathscr{H}}$ in E^3. The next projection plane in the sequence, F_2, is chosen to be orthogonal to the chord Ch_1. Thus, the chord Ch_1 projects as a single point on F_2. In the F_2 projection, the chord Ch_1 is said to be "collapsed." This procedure can be implemented as a rotation of the points in the set A about a suitable axis, denoted ℓ_R. For a given set of coordinate axis and a given projection plane F, the points in A are rotated such that the chord Ch_1 projects as a point on the projection plane F. The algorithm that is presented below uses this procedure.

The present algorithm is an adaptation of an algorithm developed by Appel and Will [26]. Appel and Will's algorithm finds the convex hull of a three-dimensional array of points by the method of "collapsed"

chords, as was discussed above. The present algorithm differs from that of [26] mainly in that the bounding problem requires knowledge of only two regions on the surface of $\hat{\mathscr{H}}$, while the algorithm given in [26] provides a description of the entire surface. The present algorithm attempts to converge quickly to the regions on the surface of $\hat{\mathscr{H}}$ of interest, at the expense of the knowledge of the over-all shape of the hull. This is appropriate because the additional information would not be relevant to the bounding problem at hand.

2.3 Problem Statement

Consider the twisted curve $\mathscr{C}$ defined in terms of the parameter u by the equation set

$$
\begin{aligned}
x &= h(u) \\
y &= g(u) \qquad u \in I = [a, b] \\
z &= f(u)
\end{aligned}
\tag{2.5}
$$

where I is the finite interval of definition of $\mathscr{C}$. The functions h(u), g(u), and f(u) are assumed to be continuous and finite valued on I. The elements of the triple (x, y, z) can be associated with the coordinate axes of a standard right handed coordinate system. The curve $\mathscr{C}$ is traced out in E^3 as u covers its interval of definition I.

Consider the line in E^3 defined by the equation set

$$
\begin{aligned}
x &= m_1 = E_U[h(u)] \\
y &= m_2 = E_U[g(u)] \\
z &= z
\end{aligned}
\tag{2.6}
$$

where the notation $E_U[\cdot]$ represents the expectation with respect to the random variable u. Let the line of equation set (2.6) be denoted by ℓ. A consequence of the isomorphism theorem [11] is that ℓ passes through the convex hull $\mathscr{H}$ generated by the curve $\mathscr{C}$. A further consequence is that the quantity

$$
m_3 = E_U[f(u)] \tag{2.7}
$$

is upper and lower bounded by the points on the surface of $\mathscr{H}$ where $\mathscr{H}$ is penetrated by the line ℓ. This amounts to an informal statement of the moment bounding theorem.

The purposes of the moment bounding technique are served by the evaluation of the surface of the hull at the two locations where the surface is penetrated by the line ℓ. Thus, when the hull $\mathscr{H}$ is approximated by the polyhedron $\hat{\mathscr{H}}$, the modified bounding problem requires only the identification of the two planar faces where the line ℓ penetrates the polyhedron $\hat{\mathscr{H}}$. The algorithm to be described identifies these faces as the result of a directed search routine. This routine uses a sequence of planar projections generated by "collapsing" a sequence of surface chords.

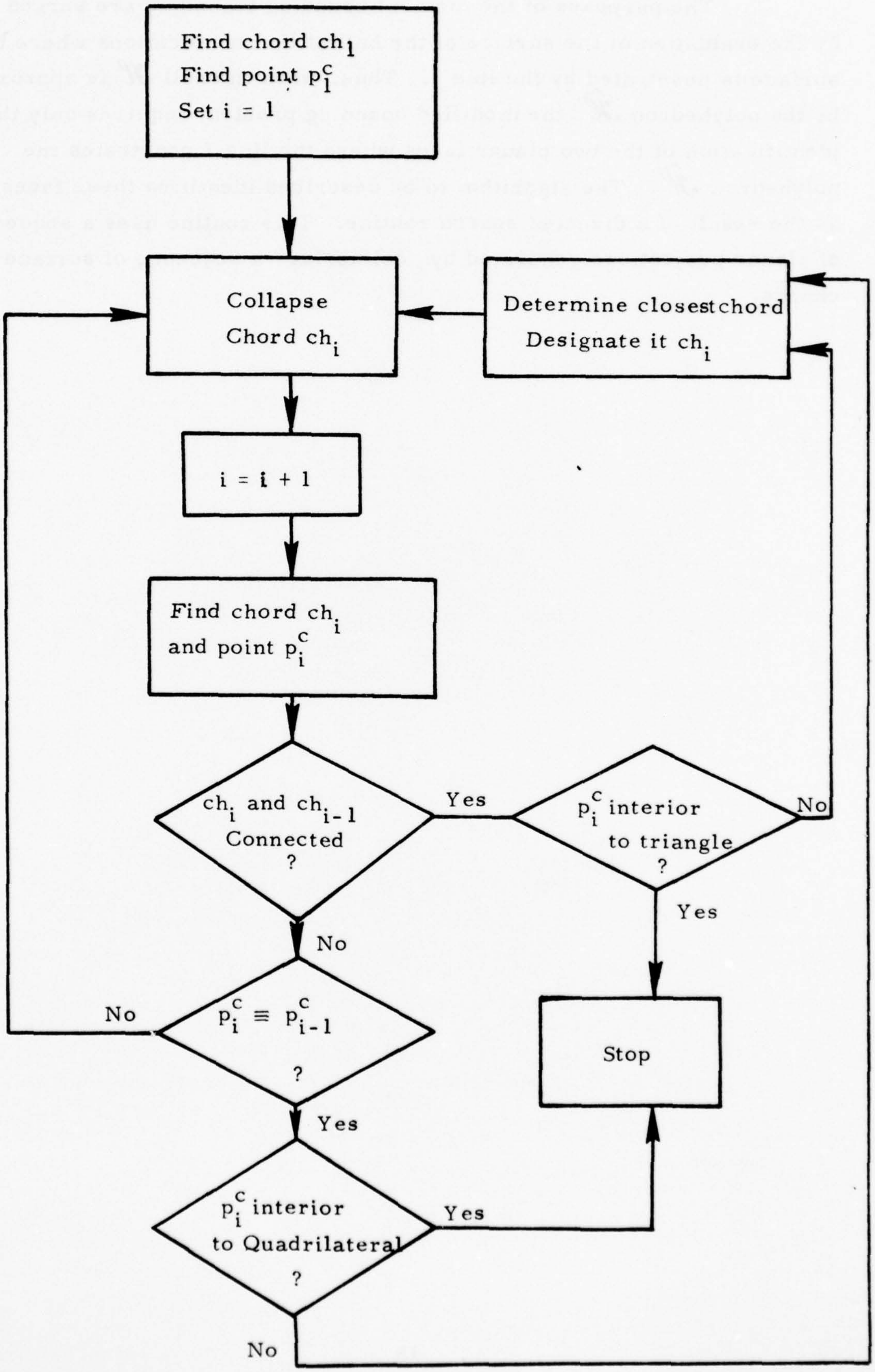

Find chord ch_1
Find point p_1^c
Set $i = 1$
Collapse
Chord ch_i
Determine closest chord
Designate it ch_i
$i = i + 1$
Find chord ch_i
and point p_i^c
ch_i and ch_{i-1}
Connected
?
Yes
p_i^c interior
to triangle
?
No
Yes
No
No
$p_i^c \equiv p_{i-1}^c$
?
Yes
Stop
p_i^c interior
to Quadrilateral
?
Yes
No

Figure 2.2

2.4 The Algorithm

This is an outline of an algorithm that computes the lower bound given by the moment bounding technique. The modifications to the algorithm that are required to compute the upper bound should be clear from the discussion. The algorithm is shown in block diagram form in Figure 2.2.

Step 1: Initialization

Given the set $A = \left\{ (x, y, z) \mid x = h(u_i),\ y = g(u_i),\ z = f(u_i);\ u_i \in I = [a, b],\ a, b < \infty,\ i = 1, \cdots, N < \infty \right\}$ and the line ℓ, given by equation set (2.6), consider the x-y plane to be the first projection plane F_1. Let ℓ_{F_1} denote the F_1 plane projection of the line ℓ. Consider the projected set A_{F_1}, the F_1 plane projection of the set A. A sketch of a possible set A_{F_1} and line ℓ_{F_1} are presented as Figure 2.3 as an aid in visualizing the situation.

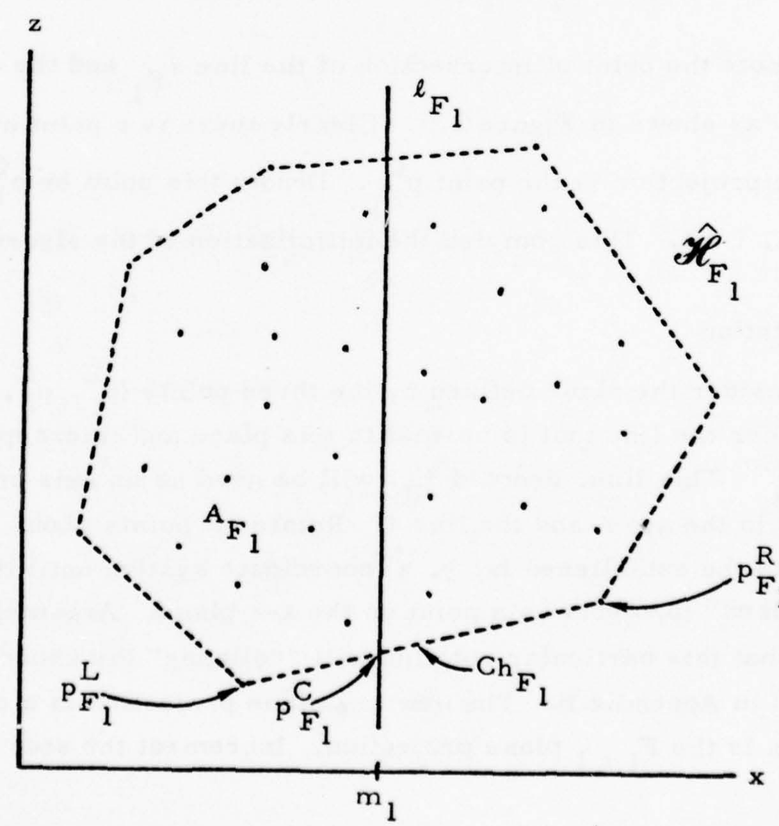


Figure 2.3

Let $\hat{\mathscr{H}}_{F_1}$ denote the convex hull of the planar set A_{F_1}. In Figure 2.3 the boundry of $\hat{\mathscr{H}}_{F_1}$ has been dashed in and designated. It can be seen that the line ℓ_{F_1} cuts the boundry of $\hat{\mathscr{H}}_{F_1}$ in exactly two places. Let Ch_{F_1} denote the chord that forms the boundry of $\hat{\mathscr{H}}_{F_1}$ at the lower of these two cuts. The chord Ch_{F_1} is designated in Figure 2.3. A subroutine for finding the chord Ch_{F_1} is described in detail in Appendix A.

Denote the left end point of Ch_{F_1} by $p^L_{F_1}$. Denote the right end point by $p^R_{F_1}$. The designations are shown in Figure 2.3. By the construction of A_{F_1}, $p^L_{F_1}$ and $p^R_{F_1}$ are projections of points in the set A. Denote these points p^L_1 and p^R_1 respectively. Let Ch_1 denote the chord connecting p^L_1 and p^R_1 in E^3. By the arguments of the previous section, since Ch_{F_1} is a chord on the surface of $\hat{\mathscr{H}}_{F_1}$, Ch_1 is a chord on the surface of $\hat{\mathscr{H}}$, the convex hull of the set A.

Denote the point of intersection of the line ℓ_{F_1} and the chord Ch_{F_1} by $p^C_{F_1}$, as shown in Figure 2.3. Clearly there is a point on the line ℓ in E^3 whose projection is the point $p^C_{F_1}$. Denote this point by p^C_1. Initialize the step index, $i = 1$. This complets the initialization of the algorithm.

Step 2: Rotation

Consider the plane defined by the three points (p^L_i, p^C_i, p^R_i) in E^3. Consider the line that is normal to this plane and intercepts the plane at the point p^C_i. This line, denoted ℓ_R, will be used as an axis of rotation for the points in the set A and the line ℓ. Rotate all points about ℓ_R with respect to the established (x, y, z) coordinate system until the chord Ch_i is "collapsed" (projects as a point on the x-z plane). Arguments establishing that this particular rotation will "collapse" the chord Ch_i are presented in Appendix B. The new x-z plane projection is a consequence of the rotation is the F_{i+1} plane projection. Increment the step index i.

Step 3: Branching

Find the next chord Ch_i by the methods of step 1. Find the next point p_i^C.

A) If the chords Ch_i and Ch_{i-1} are connected (have an end point in common), go to step 4.

B) If chords Ch_i and Ch_{i-1} are not connected, and $p_i^C \neq p_{i-1}^C$, go to step 5.

C) Otherwise, to to step 6.

Step 4: Triangular Section

The two connected chords, Ch_i and Ch_{i-1}, define a triangular planar section on the surface of $\hat{\mathscr{H}}$. If a point of the line ℓ is also a point of this triangular section, this point is the bounding point that is desired. This is the point denoted p_i^C. Terminate the procedure. If the line ℓ and the triangular section have no points in common, determine which of the three chords that bound this triangular planar section is closest to the point p_i^C in Euclidean distance. Change the labeling such that this closest chord is designated as the chord Ch_i, and its end points are designated p_i^L and p_i^R. Return to step 2. The procedures presented in this step are detailed and justified in Appendix C.

Step 5: Continue

Return to step 2 with the chord Ch_i and the point p_i^C that were found in Step 3.

Step 6: Quadrilateral

Since $p_i^C \equiv p_{i-1}^C$, both triples (p_i^L, p_i^C, p_i^R) and $(p_{i-1}^L, p_{i-1}^C, p_{i-1}^R)$ define the same plane. In this case, chords Ch_i and Ch_{i-1} are segments of the boundry of a section of this plane that is on the surface of $\hat{\mathscr{H}}$. Since Ch_i and Ch_{i-1} are co-planar but disconnected, a quadrilateral may be formed by joining appropriate pairs of end points of these chords. If the line ℓ and this quadrilateral planar section have a point in common, this point is the bounding point that is desired. This point is the point that has been denoted p_i^C. Terminate the search procedure. If the quadrilateral planar section and the line ℓ have no points in common, determine which of the four chords that form the boundry of the quadrilateral is closest in Euclidean distance to the point p_i^C. Change the labeling such that this closest chord is designated as the chord Ch_i, and its end points are designated p_i^L and p_i^R. Return to step 2. The statements are procedures presented in this step are detailed and justified in Appendix D.

2.5 Examples of Numerical Results

The algorithm outlined in Section 2.4 has been coded for general purpose computers using the FORTRAN IV programming language. A print out of the program is provided as Appendix F. Two sample runs were made using this program. The results of the first will be presented in detail in this section. The second will be presented in conjunction with the results of the four-dimensional algorithm described in the next chapter.

The examples are intersymbol interference problems. The auxillary functions used for the bounds are the second and fourth powers of the amplitude of the interference. The parametric equations for the curve $\mathscr{C}$ in E^3 are then

$$x = u^2 \tag{2.8}$$

$$y = u^4 \tag{2.9}$$

$$z = \frac{1}{2}\left[Q\left(\frac{h_o + u}{\sigma}\right) + Q\left(\frac{h_o - u}{\sigma}\right)\right] \tag{2.10}$$

where u is the amplitude of the intersymbol interference, h_o is the amplitude of the desired signal, σ is the standard deviation of the additive white Gaussian noise, and $Q(\cdot)$ is the usual complementary error function given by

$$Q(y) = \int_y^\infty \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx. \tag{2.11}$$

This is the three-dimensional extension of an intersymbol interference problem treated by Yao and Tobin [11] and Yan [12]. The range of the parameter u is

$$0 \le u \le D \le h_o \tag{2.12}$$

where D is the amplitude of the maximum interference possible. The relation (2.12) implies that the intersymbol interference "eye" is open. This means that if the channel were noise free, perfect communication would be possible inspite of the intersymbol interference. This is true because the amplitude of the desired signal, h_o, is strictly greater than the maximum interference D. This is a characteristic of a useful communication channel.

The first specific example is that of a channel with Chebychev filter frequency characteristics. This is a channel that is commonly used in comparing bounding methods in intersymbol interference channels. The characteristics of this channel are given in detail in [11] and [12]. The algorithmic results that are presented as Table 2.1 were obtained by approximating the curve $\mathscr{C}$ with an array A made up of 50 points that were equally spaced in terms of the parameter u. The exact results presented in Table 2.1 were computed using the regularity condition results presented in [34]. It was shown in [34] that the regularity condition held for the values of signal-to-noise ratio that are shown in Table 2.1 for the Chebychev channel.

It can be seen from the Table that the algorithmic values are very close to the exact values of the bounds for low and intermediate values of signal-to-noise ratio. It is noted that the algorithmic values are not typically interior to the exact bounds as was theoretically predicted. This can be attributed to cumulative round-off and truncation errors in the particular implementation of the algorithm that was used in these computations. The specific implementation that was used in obtaining the results of Table 2.1, was written to prove the validity of the concepts of the procedure. It was not optimized in terms of either cumulative computational errors or run time. In spite of this, most of the results are seen to be accurate to four significant figures. Furthermore, the typical computation required only three iterations of the algorithm to obtain convergence, and required less than 0.022 sec of central processor (CPU)

SNR dB	Exact Lower Bound [34]	Algorithmic Lower Bound	Exact Upper Bound [34]	Algorithmic Upper Bound
0	1.5931×10^{-1}	1.5931×10^{-1}	1.5931×10^{-1}	1.5931×10^{-1}
4	5.7725×10^{-2}	5.7725×10^{-2}	5.7725×10^{-2}	5.7725×10^{-2}
8	6.7573×10^{-3}	6.7574×10^{-3}	6.7578×10^{-3}	6.7578×10^{-3}
12	6.5950×10^{-5}	6.5954×10^{-5}	6.7293×10^{-5}	6.7294×10^{-5}
16	3.0333×10^{-9}	2.9472×10^{-9}	1.6576×10^{-8}	1.5539×10^{-8}
20	3.3374×10^{-19}	1.7364×10^{-19}	3.6323×10^{-15}	2.0413×10^{-14}

TABLE 2.1

time on a CDC 7600 computer. With greater attention to program optimization, in terms of run-time and computational accuracy, these statistics may be improved. However, it is clear from this example that this algorithm yields reasonably accurate results with a modest investment in computation time.

The second example is a modified version of the Chebychev channel that was examined in the first example. In the present case, the value of the maximum distortion, denoted by D in expression (2.12), is increased to three times that of the regular Chebychev channel. The channel impulse response, and other appropriate parameters are scaled accordingly. As was the case in the previous example, the second and fourth powers of the parameter u were used as the auxillary functions.

This second case is an example of a channel with severe intersymbol interference. The maximum interference D is 85% of the amplitude of the desired signal, h_o. Unfortunately, the regularity condition results presented in [34] cannot be used to compute exact bounds to this problem for all signal-to-noise ratios of interest. Therefore, a table similar to Table 2.1 cannot be presented. The results of this example are presented in Table 3.2 in Chapter III as a part of a comparison between this three-dimensional and a four-dimensional algorithm.

From the stand point of computational complexity, it is interesting to consider the effect on run time of varying the number of points contained in the array A. It can be seen from inspection of the algorithm outline (Section 2.4) and the printout of the actual program (Appendix F) that the algorithmic steps most effected by the size of the array A are the "collapsing" or rotation procedures and the procedures that find the chords Ch_F. Clearly, the complexity of the rotation procedure is linear in the number of points in A. If s seconds are required to rotate

a single point, roughly ns seconds will be required to rotate n points. The change in complexity of the chord finding routine is less clear. The routine will typically converge to the desired chord in a small number of iterations. This number will usually be essentially independent of the number of points in A. In this case, the chord finding routine would also be linearly complex in the number of points in A. However, in very poorly conditioned situations, the routine may be able to eliminate only a single point from further consideration per iteration. In this extreme case,

$$\sum_{i=2}^{N} i = \frac{N(N+1)}{2} - 1 = \frac{1}{2}(N^2 + N - 2) \tag{2.13}$$

iterations of the routine would be required to select the appropriate two points out of the N points in the array. Thus, in practice the chord finding routine may be expected to behave linearly with the number of points in the array A. In the worst possible case it would behave quadratically with the number of points. Therefore, the increase in algorithmic complexity with an increase in the number of points in the array A is manageable. A linear increase is slow enough to assure that the number of points in A can be made large enough to yield sufficiently accuracy results without causing and unreasonable increae in algorithm run time.

III. FOUR AND HIGHER DIMENSIONAL MOMENT BOUNDING ALGORITHMS

3.1 Introduction

In Chapter II an algorithmic solution to the three-dimensional moment bounding problem was demonstrated to be feasible and useful. The next obvious question would be whether an algorithmic approach could be used to compute higher dimensional bounds. Higher dimensional bounding is desirable because it promises tighter bounds than otherwise achievable. However, the evaluation of higher dimensional bounds by geometric arguments based on the Isomorphism Theorem appears to be extremely difficult. Furthermore, the evaluation of bounds of this type does not seem to have been considered. Some non-geometric approaches have been considered ([14], [15]). Thus, an algorithmic approach, such as was developed for the three-dimensional case in Chapter II, may prove to be a useful tool in computing tight bounds for general classes of auxillary functions based on higher dimensional moment bounding theory. In this chapter, an algorithm is developed that will solve the four-dimensional moment bounding problem for a very general class of auxillary functions. The algorithm has not been optimized in any computational sense, but has the advantage of relative conceptual simplicity. Furthermore, it will be shown to be easily extendable to problems of higher dimensionality than four.

3.2 Preliminary Discussion

The convex hull $\hat{\mathscr{H}}$, of an array A, containing a finite number of points in E^n is an n-dimensional polyhedron. That is, $\hat{\mathscr{H}}$ is a convex figure whose major surface features are sections of (n-1)-dimensional hyperplanes. In the four-dimensional case, these three-dimensional sections are defined by sets of four points of A. The solution to the moment bounding problem amounts to finding the two four-point sets that define the surface of the convex hull at the places where it is penetrated by the line ℓ. Except for the number of points involved in the surface definition (four instead of three), this statement of the problem is identical to that of the three-dimensional problem that was treated in Chapter II.

Unfortunately, the four-dimensional problem does not appear to yield to the same sort of intuitive solution that was used in the three-dimensional case. In particular, the notion of an orthogonal projection onto a plane (E^2) is not well defined in E^4. Therefore, chord "collapsing" in a plane is meaningless in the sense that it was used in Chapter II.

The algorithm that will be presented below makes use of two general properties of n-dimensional convex figures [22], [31] - [33]. The first of these properties is that the intersection of a convex hull and a closed half space is also a convex hull. The second property is that if a point in E^n is exterior to a convex hull in E^n, there is a (not necessarily unique) support or tangential hyperplane to the hull that separates the point and the hull into different half spaces. These two properties are used in the algorithm to separate the points in the set A into two subsets. This separation allows the algorithm to be assured of proceeding in the direction of the desired surface feature. The exact implementation of this search procedure will be detailed in the presentation of the algorithm.

3.3 The Algorithm

This section is an outline of an algorithm that solves for the lower bound for the four-dimensional moment bounding problem. The extensions of this algorithm to include the upper bound or higher dimensional problems will be apparent.

Step 1: Initialization

Given the set A and the line ℓ, consider the three-dimensional subspace of E^4 defined by the x, y, and z coordinate axes. Consider the projection of the four-dimensional convex hull $\hat{\mathcal{H}}$ onto this subspace. Denote this projection by $\hat{\mathcal{H}}_3$. Find a triangular planar section on the surface of $\hat{\mathcal{H}}_3$. This could be done using the methods of Chapter II. Denote the three points that define this planar surface feature of $\hat{\mathcal{H}}_3$ by p_1, p_2, and p_3. Continue to Step 2.

Step 2: Separation

Construct a hyperplane, to be denoted hp_d, that contains the points p_1, p_2, and p_3, and is parallel to the line ℓ. This hyperplane separates the space E^4 into two half spaces. It also separates the set A into two subsets, denoted A_ℓ and A_o. The subset A_ℓ is the subset of A that contains all the points of A that are in the same half space as the line ℓ. The subset A_o contains the remaining points of A.

Step 3: Hyperplane

Find a fourth point, p_4 such that $p_4 \in A_\ell$ and the hyperplane defined by (p_1, p_2, p_3, p_4), to be denoted hp_s, is a tangent hyperplane to the surface of $\mathcal{H}$, the convex hull of the set A. Clearly a section of hp_s will be a surface feature of $\hat{\mathcal{H}}$.

Step 4: Branching

Determine whether the line ℓ penetrates $\hat{\mathscr{H}}$ in the interior of the surface section defined by the points (p_1, p_2, p_3, p_4). If so, this point of penetration is the desired boundry point. Terminate the procedure. If not, continue to Step 5.

Step 5: Reset

Discard the point in the set $\{p_1, p_2, p_3\}$ that is "furthest" from the line ℓ. Relabel the two survivers plus the point p_4 as the new points p_1, p_2, and p_3. Return to Step 2. The form of the distance measure will be described in the discussion below.

The algorithm is presented in block diagram form in Figure 3.1. A more complete description of the steps in the algorithm is given below.

As the title implies, the purpose of the first step in the algorithm is to determine an initial position for the search routine. The search routine operates by finding a sequence of sets of four points of the array A. Each set in this sequence $\{(p_1, p_2, p_3, p_4)\}$, defines a hyperplane. Because of the method of selection of the sets, these hyperplanes will include a section of the surface of the convex hull $\hat{\mathscr{H}}$ of the array A. Having found an initial surface section, the routine operates by moving from this surface section to an adjacent section. Travel across the surface of $\hat{\mathscr{H}}$ is always toward the line ℓ. The routine continues until a surface section that is penetrated by the line ℓ is found. This point of penetration, the point of intersection of the line ℓ and the hyperplane (p_1, p_2, p_3, p_4), is the desired bounding point.

In Chapter II an argument used in the development of the three-dimensional algorithm was that the convex hull of a projection of a set of points is equivalent to the projection of the convex hull of the set (equation (2.4)). The three-dimensional set of points, A_3 formed by considering only the x, y, and z coordinate values of the four-dimensional set $A = \{(x, y, z, w)\}$, is a three-dimensional projection of A. Therefore, a surface section of

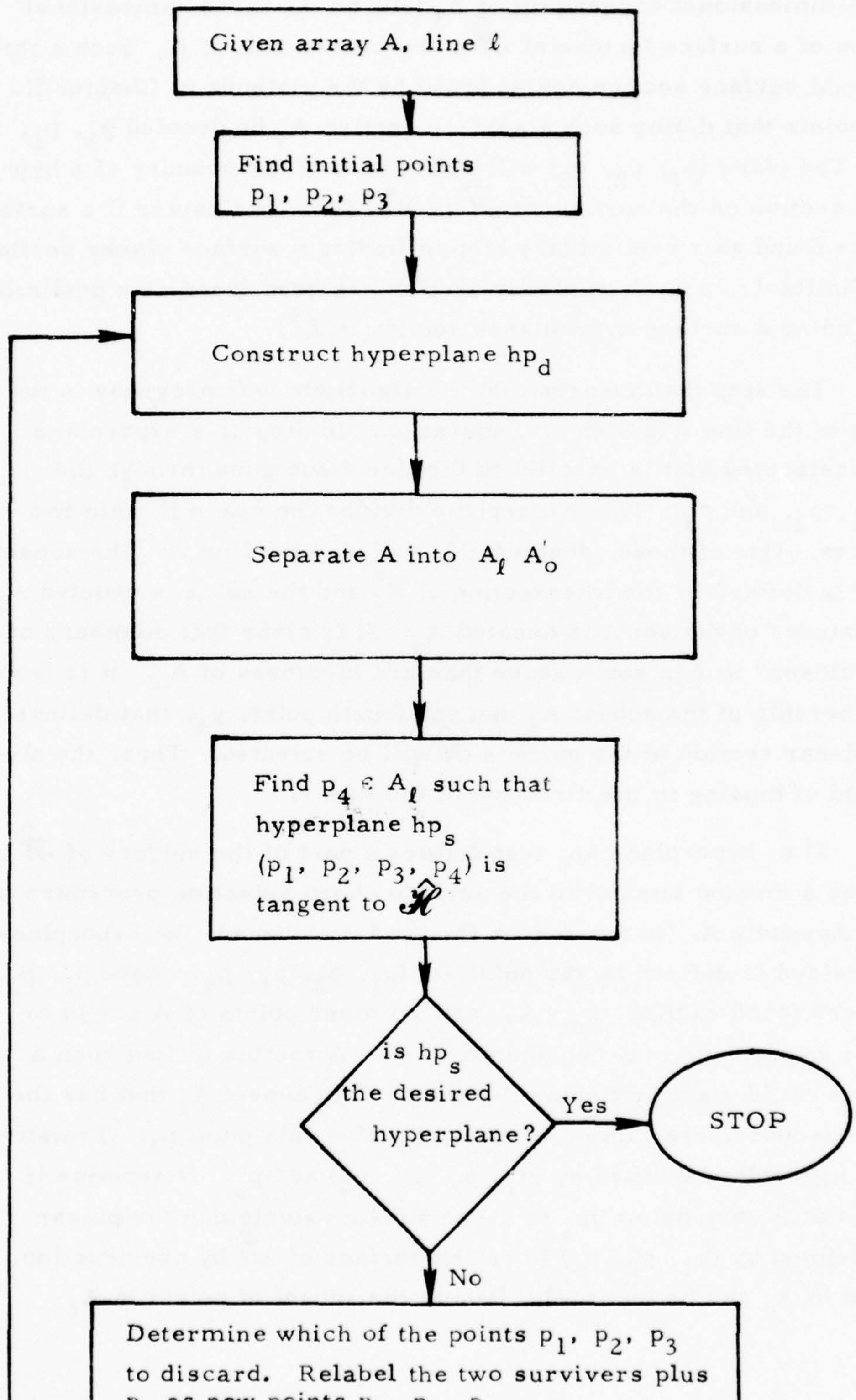

Given array A, line ℓ
Find initial points p_1, p_2, p_3
Construct hyperplane hp_d
Separate A into A_ℓ A_o'
Find $p_4 \in A_\ell$ such that hyperplane hp_s (p_1, p_2, p_3, p_4) is tangent to $\hat{\mathscr{H}}$
is hp_s the desired hyperplane?
Yes
STOP
No
Determine which of the points p_1, p_2, p_3 to discard. Relabel the two survivers plus p_4 as new points p_1, p_2, p_3.

FIGURE 3.1

the three-dimensional convex hull of A_3 will be the three-dimensional projection of a surface feature of $\hat{\mathscr{H}}$, the convex hull of A. Such a three-dimensional surface section can be found by the methods of Chapter II. Let the points that define such a surface section A_3 be denoted p_1, p_2, and p_3. The plane (p_1, p_2, p_3) will define part of the boundry of a hyperplanar section on the surface of $\hat{\mathscr{H}}$ in E^4 [30]. In Chapter II a surface chord was found as a preliminary step to finding a surface planar section in E^3. Similarly, a surface planar section has been found as a preliminary step to finding a surface hyperplanar section in E^4.

The step that assures that the algorithm will progress in the direction of the line ℓ is Step 2: Separation. In Step 2, a hyperplane hp_d is constructed that is parallel to the line ℓ and goes through the points p_1, p_2, and p_3. This hyperplane divides the space E^4 into two half spaces. One of these, denoted E^4_ℓ, includes the line ℓ. The subset of A that is defined by the intersection of E^4_ℓ and the set A is denoted A_ℓ. The remainder of the set A is denoted A_o. It is clear that members of A_ℓ are "closer" to ℓ in some sense than are members of A_o. It is from the membership of the subset A_ℓ that the fourth point, p_4, that defines a hyperplanar section of the surface $\hat{\mathscr{H}}$ will be selected. Thus, the algorithm is assured of moving in the direction of the line ℓ.

The hyperplane hp_s that defines a part of the surface of $\hat{\mathscr{H}}$ is found by a method similar to the surface chord selection procedure introduced in Appendix A. In the search for the lower bound, the hyperplane that is desired is defined by the point set (p_1, p_2, p_3, p_4) where p_1, p_2, and p_3 were found earlier, $p_4 \in A_\ell$, and all other points of A are in or above this hyperplane in z-coordinate value. A routine to find such a hyperplane could start by finding the point in the subset A_ℓ that has the minimum z-coordinate value. Tentatively label this point p_4. Tentatively label the hyperplane defined by (p_1, p_2, p_3, p_4) as hp_s. Determine if any points of A_ℓ are below hp_s in z-coordinate value(since the planar section defined by (p_1, p_2, p_3) is on the surface of $\hat{\mathscr{H}}$ by construction, the points in A_o can be ignored). Denote the subset of points in A_ℓ

that lie below hp_s by B_ℓ. If B_ℓ is empty, hp_s is the desired hyperplane. If B_ℓ is not empty, find the point in B_ℓ with the minimum z-coordinate value. Tentatively label this new point p_4. This defines a new tentative hyperplane hp_s to be tested. It is clear that this routine will converge to the desired hyperplane from the arguments presented in Appendix A.

The remaining step in the algorithm is the reset step. In this step three of the four points (p_1, p_2, p_3, p_4) are selected to form the basis for the search for the next hyperplane. These will be the three points that are "closest" to the line ℓ in a special sense.

Let the point of intersection between the hyperplane hp_s defined by (p_1, p_2, p_3, p_4), and the line ℓ be denoted by p_ℓ. Consider the plane defined by (p_2, p_3, p_4) and the line defined by (p_ℓ, p_1). Since both this line and this plane are within the hyperplane hp_s, they will intersect (a line and a plane do not intersect in general in E^4 [30]). Let d_1 denote the Euclidean distance between the point p_ℓ and the point of intersection between the line and the plane. Similarly, consider the plane (p_1, p_3, p_4) and the line (p_ℓ, p_2). Let d_2 be the distance between p_ℓ and the intersection of this new line and plane. A distance d_3 can be similarly defined using the plane (p_1, p_2, p_4) and the line (p_ℓ, p_3). The plane selected for the next iteration of the search procedure is the plane associated with the minimum value element in the set (d_1, d_2, d_3). It is noted that the combination of the plane (p_1, p_2, p_3) and the line (p_ℓ, p_4) need never be considered. This is because the method of selection of the point p_4 assures that the closest plane will be one of those with the point p_4 as a part of its definition. Infact, if p_4 was eliminated at this step, it would cause the algorithm to cycle endlessly.

This constitutes the essence of a demonstration that the algorithm will converge in a finite number of iterations. The algorithm considers only surface sections, of which there are a finite number, and is always moving in the direction of the line ℓ. Thus, the algorithm must converge to the proper surface feature in a finite number of iterations.

3.4 Examples of Numerical Results

The algorithm outlined in Section 3.3 has been coded for general purpose computers using the FORTRAN IV programming language. A printout of the program is provided in Appendix G. Two sample runs were made using this program. These runs are the four-dimensional extensions of the sample runs presented in Section 2.5 for the three-dimensional algorithm.

The examples are intersymbol interference problems. The auxillary functions used in obtaining the bounds were the second, fourth, and sixth powers of the amplitude of the interference. The parametric equations for the curve $\mathscr{C}$ are

$$x = u^2 \tag{3.1}$$

$$y = u^4 \tag{3.2}$$

$$w = u^6 \tag{3.3}$$

$$z = \frac{1}{2}\left[Q\left(\frac{h_o + u}{\sigma}\right) + Q\left(\frac{h_o - u}{\sigma}\right)\right] \tag{3.4}$$

where u is the value of the amplitude of the intersymbol interference, h_o is the amplitude of the desired sign, σ is the standard deviation of the additive white Gaussian noise, and $Q(\cdot)$ is the usual complementary error function given by

$$Q(y) = \int_y^\infty \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx. \tag{3.5}$$

As was the case in Chapter II, the limits on the value of u are given by

$$0 \leq u \leq D < h_o, \tag{3.6}$$

where D represents the maximum possible amplitude of the intersymbol interference.

As in Chapter II, the first specific example is that of a channel with a Chebychev filter impulse response. The results that are presented in Table 3.1 were obtained by approximating the curve $\mathscr{C}$, given parametrically by equations (3.1) - (3.4), by an array A containing 50 points. These points were equally spaced in terms of the parameter u.

Under the general heading "Four-Dimensional Bounds," Table 3.1 contains the upper and lower bounds that were computed using the computer program shown in Appendix G. Also shown under this heading is the difference between these bounds. The difference serves as a measure of the tightness of the four-dimensional bounding technique. For purposes of comparison, the results computed with the three-dimensional routine (Appendix F) that were presented in Section 2.5 are reproduced in Table 3.1. As expected, the four-dimensional upper bounds are lower than the three-dimensional upper bounds. Also, the four-dimensional lower bounds are higher than the three-dimensional lower bounds. This is reflected in the difference columns. The four-dimensional bounds are seen to be as much as four orders of magnitude tighter than the three-dimensional bounds, and typically about two orders of magnitude tighter. Thus, there is a significant improvement in the tightness of the bounds that can be expected from the additional dimension. The corresponding drawback is in runtime. The four-dimensional algorithm of Appendix G required roughly two and a half times as much processing time as did the three-dimensional algorithm of Appendix F. The algorithm presented in Appendix G (the four-dimensional algorithm) is not optimized in terms of run time. Nevertheless, one is lead to expect that a sizeable processing time penalty may be required for each additional dimension used in the bounding.

The second example is that of a channel with a modified version of a Chebychev impulse response. In this example, the maximum distortion, D in expression (3.6), is taken to be three times that of the standard Chebychev channel. The channel impulse response and the other appropriate parameters are scaled accordingly. This example corresponds to the second example discussed in Section 2.5.

SNR	FOUR-DIMENSIONAL BOUNDS			THREE-DIMENSIONAL BOUNDS		
	Upper Bound	Lower Bound	Diff.	Upper Bound	Lower Bound	Diff.
dB	B_U	B_L	$B_U - B_L$	B_U	B_L	$B_U - B_L$
0	1.5931×10^{-1}	1.5931×10^{-1}	5.4907×10^{-12}	1.5931×10^{-1}	1.5931×10^{-1}	1.7026×10^{-8}
4	5.7725×10^{-2}	5.7725×10^{-2}	3.3281×10^{-10}	5.7725×10^{-2}	5.7725×10^{-2}	5.3433×10^{-8}
8	6.7577×10^{-3}	6.7577×10^{-3}	1.7327×10^{-9}	6.7578×10^{-3}	6.7574×10^{-3}	4.7458×10^{-7}
12	6.6205×10^{-5}	6.6198×10^{-5}	7.0937×10^{-9}	6.7294×10^{-5}	6.5954×10^{-5}	1.3405×10^{-6}
16	5.5281×10^{-9}	3.8831×10^{-9}	1.6451×10^{-9}	1.5539×10^{-8}	2.9472×10^{-9}	1.2591×10^{-8}
20	3.8729×10^{-16}	1.3277×10^{-18}	3.8596×10^{-16}	2.0413×10^{-14}	1.7364×10^{-19}	2.0413×10^{-14}

TABLE 3.1

The numerical results are presented in Table 3.2. The results in this table are presented in the same format as the results presented in Table 3.1. For this example of more extreme amounts of intersymbol interference, the four-dimensional bounds can be seen to be about an order of magnitude tighter than the three-dimensional bounds, over the range of signal-to-noise ratios. Thus, this example indicates that there is still a significant advantage to be gained from the higher dimensional bounds for the case of severe intersymbol interference.

SNR	FOUR-DIMENSIONAL BOUNDS			THREE-DIMENSIONAL BOUNDS		
	Upper Bound	Lower Bound	*Diff.*	Upper Bound	Lower Bound	Diff.
dB	B_U	B_L	$B_U - B_L$	B_U	B_L	$B_U - B_L$
0	1.6445×10^{-1}	1.6445×10^{-1}	3.5336×10^{-8}	1.6446×10^{-1}	1.6445×10^{-1}	5.1784×10^{-6}
4	6.7405×10^{-2}	6.7403×10^{-2}	1.7210×10^{-6}	6.7410×10^{-2}	6.7384×10^{-2}	2.5611×10^{-5}
8	1.4003×10^{-2}	1.4001×10^{-2}	2.5350×10^{-6}	1.4071×10^{-2}	1.3715×10^{-2}	3.5605×10^{-4}
12	1.2972×10^{-3}	1.2113×10^{-3}	8.5834×10^{-5}	1.6913×10^{-3}	1.1126×10^{-3}	5.7866×10^{-4}
16	1.7687×10^{-4}	3.0458×10^{-5}	1.4642×10^{-4}	8.1192×10^{-4}	6.4553×10^{-6}	8.0547×10^{-4}
20	6.7317×10^{-5}	6.2781×10^{-9}	6.7310×10^{-5}	3.1544×10^{-4}	3.0980×10^{-11}	3.1544×10^{-4}

TABLE 3.2

3.5 Conclusions

The results presented in Section 3.4 demonstrate the value of higher dimensional bounding routines. This is that considerably tighter bounding results can be obtained for some additional expense in the form of computer run time. In the results shown in Section 3.4, the bounds were typically one or two orders of magnitude tighter while computation time rose by a factor of less than three.

These results also demonstrate the efficiency of the four-dimensional algorithm. This algorithm systematically stepped across the surface of the convex hull until the appropriate surface feature was found. This was accomplished by considering a sequence of hyperplanar surface features in terms of the sets of four points that define them. The algorithm proceeded by identifying and eliminating the point out of the four point set that was "furthest" from the desired direction of travel. The eliminated point was then replaced by an appropriate point that was in the direction of convergence. This new four point set defined another hyperplanar surface feature that was "closer" to the solution of the bounding problem than was the previous section. This procedure was repeated until convergence to the solution of the bounding problem was achieved.

The extension to bounding problems of dimensions greater than four seems clear. For a five-dimensional problem the hyperplanes would be defined by sets of five points. In a manner similar to the four-dimensional algorithm, one of these points that is "furthest" from the desired direction of travel could be identified and eliminated. This point could be replaced with an appropriate point that is "closer" to the solution of the bounding problem. This would define a new hyperplanar surface reaction that is closer to the solution of the problem. Clearly, if this procedure is repeated a sufficient number of times, the algorithm will converge to the desired solution. It is also clear that the concept of stepping across the surface of a convex hull by modifying the set of points one point at a time is a concept that will work independently from the total number of points in the set. That is, the procedure of stepping across the surface by systematically going from one surface feature to an adjacent on is a procedure that will work independent of the dimensionality of the problem.

An important point to investigate is the relationship between problem dimension and algorithm runtime. This is a difficult problem, but some insight can be gained from the form of the four-dimensional algorithm.

In order to determine whether this desired solution has been found, the algorithm must compute the point of intersection of a hyperplane and the line ℓ. In E^4, this amounts to inverting a 4 x 4 matrix. In an extension to a K-dimensional problem, it would mean inverting a K x K matrix. The niave method of inverting a K x K matrix (straightforward use of the method of cofactors) would require more than 2(K!) multiplications. While more sophisticated numerical methods would undoubtedly require fewer multiplications, it appears that run time can be expected to increase very quickly as the dimensionality of the problem increases. Fortunately, the numerical results presented here appear to indicate the extensions to very high dimensionality will rarely be necessary. This is because the bounding results appear to tighten very quickly with increasing problem dimension.

Appendix A

Algorithm

An algorithm for finding the chord Ch_F as required in Step 1 and Step 3 is given below.

a) Divide the set A_F into two subsets. Denote these subsets A_F^L and A_F^R. Assign to A_F^L all points of A_F that are to the left of the line ℓ_F. Assign all remaining points to the set A_F^R.

b) In each of the sets A_F^L and A_F^R, find the point whose z-coordinate value is the minimum. If the minimum z-coordinate value in either or both subsets is not unique, select the minimum point or points that are closest to the line ℓ_F. Designate the resulting point in A_F^L by p_F^L, and the point in A_F^R by p_F^R.

c) Consider the line in the plane F defined by the two points p_F^L and p_F^R. If there are no points of A_F below this line, the chord defined by p_F^L and p_F^R is the desired chord Ch_F. In this case, terminate the search. If there are points of A_F below this line, contine to d).

d) Consider the points of the set A_F that are below the line defined by the points p_F^L and p_F^R. Denote these points as the set B_F. Let B_F^L denote the intersection of the sets B_F and A_F^L. Similarly, let B_F^R denote the intersection of the sets B_F and A_F^R. Find the point in the set B_F^L whose z-coordinate value is the minimum. Let this new point assume the designation p_F^L. If B_F^L is empty, the designation p_F^L remains unaltered. Similarly, find the point in the set B_F^R whose z-coordinate value is the minimum. Let this point assume the designation p_F^R. If B_F^R is empty, the designation p_F^R remains unaltered. Return to c) with the modified pair of points (p_F^L, p_F^R).

Discussion

This algorithm is demonstrated to terminate with the proper chord in a finite number of steps by the following argument:

Let ℓ_c stand for the line and Ch_c stand for the chord defined by the pair of points (p_F^L, p_F^R) that are considered at step c) of the algorithm. Similarly, let ℓ_d stand for the line and Ch_d stand for the chord defined by the new pair of points found in step d) of the algorithm. If the algorithm did not continue to step d), then all of the points of A_F are on or above the line ℓ_c. This means that the chord Ch_c is on the boundry of $\hat{\mathcal{H}}_F$, the convex hull of A_F [22]. But by construction, Ch_c is intercepted by the line ℓ_F. Therefore, Ch_c is the chord that is desired. The chord Ch_c will be labeled Ch_F by the algorithm at step c) and the algorithm will terminate. If the algorithm continues to step d), then there are points of A_F that lie below the line ℓ_c. In this case, at least one of the end points of the chord Ch_d must be below the line ℓ_c. But then the interior of the chord Ch_d must be strictly below the interior of the chord Ch_c. In particular, the point at which Ch_d intercepts the line ℓ_F, denoted p_d, must be lower in z-coordinate value than the intercept between ℓ_F and Ch_c, denoted p_c. These ideas are illustrated in Figure A. 1.

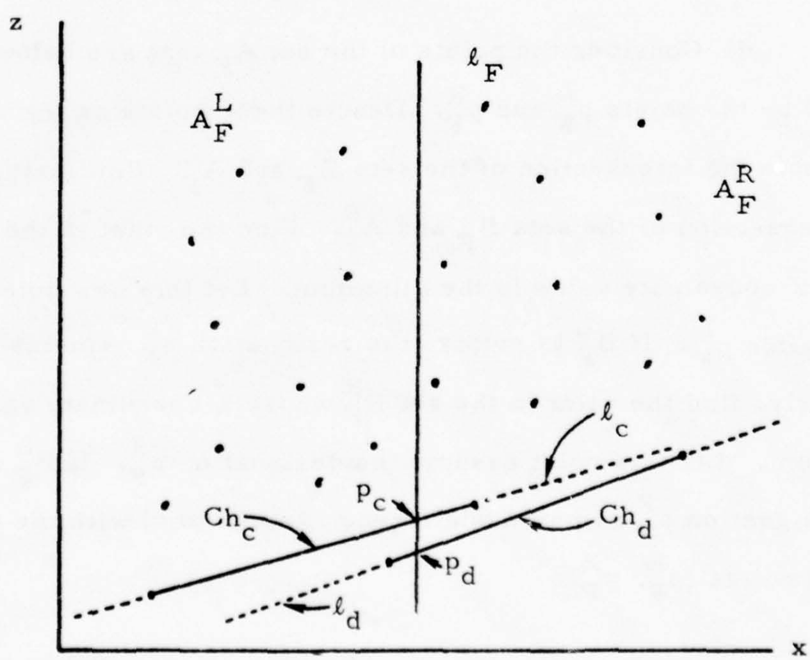


Figure A. 1

Thus, with each pass through step d), the point p_d must be lowered in z-coordinate value. This means that this algorithm cannot cycle back to old pairs of points (p_F^L, p_F^R). This is because the z-coordinate value of the intercept point, p_d, is strictly decreasing. Since the set A_F has a finite number of elements, the search must terminate in a finite number of operations. It would be expected that the termination would typically occur after a small number of iterations.

Appendix B

By construction, the three points p_i^L, p_i^C, and p_i^R, project onto the plane F_i as three points on the line which has the chord Ch_{F_i} as a line segment. Therefore, all points in the plane defined by (p_i^L, p_i^C, p_i^R) will project onto the plane F_i as points on this line. This implies that the plane (p_i^L, p_i^C, p_i^R) is normal to the plane F_i.

If a plane is rotated about an axis normal to itself, the orientation of the plane in E^3 remains unchanged. The only effect is to alter the positions of the points within the plane relative to a fixed coordinate system. In particular, the plane (p_i^L, p_i^C, p_i^R) can be rotated about a line normal to it until the chord Ch_i is normal to the plane F_i. Such a rotation must be possible since the rotation will not effect the normality of the planes with respect to each other. When the chord Ch_i is normal to the plane F_i, it will project as a single point on F_i. This new orientation of the points of the set A will yield the new projection denoted F_{i+1}, and the chord Ch_i will be said to have been "collapsed".

Appendix C

By construction, the chord Ch_{i-1} is "collapsed" in the projection F_i. Since all points of the chord Ch_{i-1} project as a single point on F_i, and since Ch_i and Ch_{i-1} share an end point, the plane defined by these two connected chords projects on the plane F_i as the line of which the chord Ch_{F_i} is a line segment. But by construction, all points of the set A lie in or above this plane. This means that the plane defined by the chords Ch_i and Ch_{i-1} is a support plane [22] of $\hat{\mathscr{H}}$, the convex hull of the set A. This plane defines part of the surface of $\hat{\mathscr{H}}$. Thus, the two connected chords Ch_i and Ch_{i-1} define a triangular planar section on the surface of $\hat{\mathscr{H}}$.

The line ℓ will intersect the surface of $\hat{\mathscr{H}}$ in two points. This is a consequence of the Isomorphism Theorem. One of these points will determine the value of the upper bound, and the other will determine the value of the lower bound. If the line ℓ and the surface triangular planar section defined by Ch_i and Ch_{i-1} have a *point in common*, it must be one of these two bounding points. Because of procedure that was used to select the chords Ch_i and Ch_{i-1}, it will be the lower bound.

It can be seen that the plane defined by the chords Ch_i and Ch_{i-1} and the plane defined by the points (p_i^L, p_i^C, p_i^R) both project as the same line on the plane F_i. Thus, these planes are identical. Since the point p_i^C is a point of the line ℓ and also a point of the support plane, if it lies within the triangular planar section defined by Ch_i and Ch_{i-1}, it must be the bounding point that is desired.

Appendix D

If $p_i^C \equiv p_{i-1}^C$, both chords Ch_i and Ch_{i-1}, are parts of the boundry of the same planar face on the surface of $\hat{\mathscr{H}}$. This statement is justified by the arguments given below.

Consider the plane defined by $(p_{i-1}^L, p_{i-1}^C, p_{i-1}^R)$. By construction, all points of A are either in or above this plane. In the projection F_i, this plane projects as the line $(p_{F_{i-1}}^L, p_{F_{i-1}}^C)$. Thus, in the F_i projection, all points of A_{F_i} must be on or above the line $(p_{F_{i-1}}^L, p_{F_{i-1}}^C)$. In particular, the points of the chord Ch_{F_i} must be on or above the line $(p_{F_{i-1}}^L, p_{F_{i-1}}^C)$. But clearly the point $p_{F_{i-1}}^C$ is on the line $(p_{F_{i-1}}^L, p_{F_{i-1}}^C)$. Thus, the point $p_{F_i}^C$ must be equal to or above (in z-coordinate value) the point $p_{F_{i-1}}^C$.

If either of the points $p_{F_i}^L$ and $p_{F_i}^R$ that define the chord Ch_{F_i} are above the line $(p_{F_{i-1}}^L, p_{F_{i-1}}^R)$, the point $p_{F_i}^C$ must be above the point $p_{F_{i-1}}^C$. Thus, if $p_{F_{i-1}}^C \equiv p_{F_i}^C$, Ch_{F_i} must be on the line $(p_{F_{i-1}}^L, p_{F_{i-1}}^C)$. But then the planes $(p_{i-1}^L, p_{i-1}^C, p_{i-1}^R)$ and (p_i^L, p_i^C, p_i^R) will both project as the same line on F_i. Thus, these planes will be identical. Therefore, by arguments similar to those of Appendix C, Ch_i and Ch_{i-1} will be chords on the boundry of the same planar face on the surface of $\hat{\mathscr{H}}$.

Appendix E

The arguments presented below show that the algorithm will converge to the appropriate face of $\hat{\mathscr{H}}$.

As was established in Appendix D, the sequence of points $\left\{ p_i^C \right\}$ is non-decreasing. Thus, the algorithm cannot cycle. Since it can never return to previously considered faces of $\hat{\mathscr{H}}$, and since $\hat{\mathscr{H}}$ has a finite number of faces, the algorithm must converge to the appropriate face in a finite number of steps.

Appendix F

This Appendix contains a listing of a FORTRAN IV computer program that solves the three-dimensional moment bounding problem. This program is based on the algorithm described in Chapter II.

```
      DIMENSION AX(100),AY(100),AZ(100)                                  00010
      COMMON N,AX,AY,AZ,PCX,PCY,PCZ,NP1,NP2,NP3,SCALE                    00020
C                                                                        00030
C           THIS PROGRAM COMPUTES UPPER AND LOWER BOUNDS TO THE FIRST    00040
C     GENERALIZED MOMENT OF A FUNCTION REPRESENTED BY THE ARRAY AZ(K).   00050
C     THE BOUNDS ARE GENERATED BY A THREE DIMENSIONAL APPLICATION OF     00060
C     AN ISOMORPHISM THEOREM FROM GAME THEORY.  THE APPROXIMATING        00070
C     FUNCTIONS ARE REPRESENTED BY THE ARRAYS AX(K) AND AY(K).  THE      00080
C     BOUNDING POINT IS GIVEN BY (AM1,AM2).                              00090
C                                                                        00100
      N=      50                                                         00110
      H=1.                                                               00120
      DB=-1.
      SIG=0.
      AM1=.07379**2                                                      00140
      AM2=.09547**4                                                      00150
      D=.28334                                                           00160
      WRITE(6,13) N,SIG,AM1,AM2,D                                        00170
   13 FORMAT(1H1,3HN =,I4,5X,5HSIG =,F10.5,5X,5HAM1 =,1PE10.3,           00180
     1     5X,5HAM2 =,1PE10.3,5X,3HD =,0PF8.5,////)                      00190
      DU=D/FLOAT(N-1)                                                    00200
      DO 200 NDB=1,21
      SIG=10.**(-DB/20.)
      AZMAX=-1.E12                                                       00220
      AZMIN=1.E12                                                        00230
      U=-DU                                                              00210
      DO 1 K=1,N                                                         00240
      U=U+DU                                                             00250
      AX(K)=U**2                                                         00260
      AY(K)=U**4                                                         00270
      A1=(H+U)/(SQRT(2.)*SIG)
      A2=(H-U)/(SQRT(2.)*SIG)
      E1=CHEBERF(A1)
      E2=CHEBERF(A2)
      IF(A1.LE.4.) E1=1.-E1
      IF(A2.LE.4.) E2=1.-E2
      AZ(K)=(E1+E2)/4.
      AZMAX=AMAX1(AZMAX,AZ(K))                                           00290
      AZMIN=AMIN1(AZMIN,AZ(K))                                           00300
    1 CONTINUE                                                           00330
      SCALE=1./(AZMAX-AZMIN)
      DO 100 K=1,N
      AX(K)=AX(K)*SCALE
      AY(K)=AY(K)*SCALE
```

```
      AZ(K)=(AZ(K)-AZMIN)*SCALE
  100 CONTINUE
C
C     DEFINE LINE L BY ITS END POINTS - INDEXED NP1 AND NP2.
C
      NP1=N+1
      AX(NP1)=AM1*SCALE
      AY(NP1)=AM2*SCALE
      AZ(NP1)=2.
      NP2=N+2.
      AX(NP2)=AM1*SCALE
      AY(NP2)=AM2*SCALE
      AZ(NP2)=-1.
      NP3=N+3.
      AX(NP3)=AM1*SCALE
      AY(NP3)=AM2*SCALE+1.
      AZ(NP3)=AZ(NP2)
C
C     INITIALIZE ROUTINE FOR LOWER BOUND
C
      ICNT=0
      ICNTL=0
      PCX=AX(NP2)
      PCY=AY(NP2)
      PCZ=AZ(NP2)
      S=1.
   10 CONTINUE
C
C     FIND BOUNDING CHORD FOR THIS INITIAL PROJECTION
C
      CALL CHORD(S,I,J)
    5 CONTINUE
      NLP1=I
      NRP1=J
C
C     COLLAPSE CHORD
C        ,
      CALL XYROT(I,J)
      ICNT=ICNT+1
      IF(ICNT.EQ.51) STOP
      PCX1=PCX
      PCY1=PCY
      PCZ1=PCZ
C
```

```
C     FIND NEW CHORD
C
      CALL CHORD(S,I,J)
C
C     CHECK FOR BOUNDRY
C
      IF(I.NE.NLP1.AND.I.NE.NRP1.AND.J.NE.NLP1.AND.J.NE.NRP1) GOTO2
      CALL TRI(I,J,NLP1,NRP1,AL1,AL2)
      IF(AMIN1(AL1,AL2,(1.-AL1-AL2)).LT.0.) GOTO 3
      IF(AMAX1(AL1,AL2,(1.-AL1-AL2)).LE.1.) GOTO 4
    3 CONTINUE
      CALL AUX(I,J,NLP1,NRP1)
      GO TO 5
    2 CONTINUE
      PD2=PCX**2+PCY**2+PCZ**2
      DPD2=(PCX1-PCX)**2+(PCY1-PCY)**2+(PCZ1-PCZ)**2
      IF(DPD2-1.E-12*PD2) 6,6,5
    6 CONTINUE
      CALL QUAD(I,J,NLP1,NRP1,AL1,AL2)
      IF(AMIN1(AL1,AL2,(1.-AL1-AL2)).LT.0.) GOTO 7
      IF(AMAX1(AL1,AL2,(1.-AL1-AL2)).LE.1.) GOTO 4
    7 CONTINUE
      CALL AUX(I,J,NLP1,NRP1)
      GO TO 5
    4 CONTINUE
C
C     BOUNDRY FOUND
C
      CALL BND(AZMAX,AZMIN,B)
      IF(S) 8,9,9
    9 CONTINUE
C
C     REINITIALIZE FOR UPPER BOUND
C
      ICNTL=ICNT
      ICNT=0
      S=-1.
      BL=B
      GO TO 10
    8 CONTINUE
      DB=B-BL
      NDBO=NDB-1
      WRITE(6,11) NDBO,B,BL,DB,ICNT,ICNTL
   11 FORMAT(1H0,I10,1P3E16.4,2I10)
```

```
  200 CONTINUE
      STOP
      END
      SUBROUTINE CHORD(S,I,J)
      DIMENSION AX(100),AY(100),AZ(100)
      DIMENSION LL(100),LR(100)
      COMMON N,AX,AY,AZ,PCX,PCY,PCZ,NP1,NP2,NP3
C
C           THIS SUBROUTINE FINDS THE BOUNDING CHORD CH. FOR S POSITIVE,
C     IT FIND THE LOWER BOUND, WHILE FOR S NEG. IT FIND THE UPPER BOUND.
C
      IF(AX(NP1).EQ.AX(NP2)) GOTO 1
      CALL XZROT(NP1,NP2)
    1 CONTINUE
      LCL=0
      LCR=0
      IF(S) 21,22,22
   22 CONTINUE
      PL=AZ(NP1)
      PR=AZ(NP1)
      I=NP1
      J=NP1
      GOTO 10
   21 CONTINUE
      PL=AZ(NP2)
      PR=AZ(NP2)
      I=NP2
      J=NP2
   10 CONTINUE
      DO 2 K=1,N
C
C     SPLIT INTO LEFT AND RIGHT OF LINE L - FIRST PASS
C
      IF(AX(K)-PCX) 3,4,4
    3 CONTINUE
      LCL=LCL+1
      LL(LCL)=K
C
C     BELOW (ABOVE) LINE L
C
      IF(S*(AZ(K)-PL)) 5,7,6
    5 CONTINUE
      I=K
      PL=AZ(K)
```

```
      GOTO 6
    7 CONTINUE
      IF(AX(K)-AX(I)) 6,6,5
    4 CONTINUE
      LCR=LCR+1
      LR(LCR)=K
      IF(S*(AZ(K)-PR)) 8,9,6
    8 CONTINUE
      J=K
      PR=AZ(K)
      GOTO 6
    9 CONTINUE
      IF(AX(K)-AX(J)) 8,6,6
    6 CONTINUE
    2 CONTINUE
      IF(LCL.NE.0.AND.LCR.NE.0) GOTO11
      WRITE(6,20)
   20 FORMAT(1H0,30HLINE L DOES NOT INTERSECT HULL)
      STOP
   11 CONTINUE
C
C     FIND POINT PC
C
      DCZ=PL-PR
      DLY=AY(NP1)-AY(NP2)
      DLX=AX(NP1)-AX(NP2)
      DLZ=AZ(NP1)-AZ(NP2)
      DCX=AX(I)-AX(J)
      F1=DLX*DCZ
      F2=DLZ*DCX
      PCZ=(PR*F2+(AZ(NP1)*AX(NP2)-AX(NP1)*AZ(NP2)-AX(J)*DLZ)*
     1     DCZ)/(F2-F1)
      PCX=(PCZ-AZ(NP2))*DLX/DLZ+AX(NP2)
      PCY=(PCZ-AZ(NP2))*DLY/DLZ +AY(NP2)
C
C     LEVEL CHORD CH
C
      CALL XZROT(I,J)
C
C     INITIALIZE AND BEGIN CHORD CHECK SEQUENCE
C
      PT=PCZ
      KL=I
      KR=J
```

```
      PTL=PT                                                            01930
      PTR=PT                                                            01940
      DO 14 M=1,LCL
      K=LL(M)
      IF(S*(AZ(K)-PTL)) 18,13,13                                        02010
   18 CONTINUE                                                          02020
      PTL=AZ(K)                                                         02030
      KL=K                                                              02040
   13 CONTINUE                                                          02110
   14 CONTINUE                                                          01980
   16 CONTINUE                                                          02060
      DO 15 M=1,LCR
      K=LR(M)
      IF(S*(AZ(K)-PTR)) 19,12,12
   19 CONTINUE                                                          02080
      PTR=AZ(K)                                                         02090
      KR=K                                                              02100
   12 CONTINUE                                                          02120
   15 CONTINUE                                                          02000
      IF(KL.EQ.I.AND.KR.EQ.J) GOTO 17                                   02130
      I=KL                                                              02140
      J=KR                                                              02150
      PL=PTL                                                            02160
      PR=PTR                                                            02170
      GOTO 11                                                           02180
   17 CONTINUE                                                          02190
      RETURN                                                            02200
      END                                                               02210
      SUBROUTINE TRI(I,J,NLP1,NRP1,AL1,AL2)                             02220
      DIMENSION AX(100),AY(100),AZ(100)                                 02230
      COMMON N,AX,AY,AZ,PCX,PCY,PCZ,NP1,NP2,NP3                         02240
C                                                                       02250
C           THIS SUBROUTINE DETERMINES THE POSITION OF THE POINT OF     02260
C     INTERSECTION BETWEEN LINE L AND THE PLANE FORMED BY THE THREE     02270
C     INDEPENDENT POINTS IN THE ARGUMENT LIST.                          02280
C                                                                       02290
      N1=NLP1                                                           02300
      N2=NRP1                                                           02310
C                                                                       02320
C     FIND INDEPENDENT POINT                                            02330
C                                                                       02340
      IF(NLP1.EQ.I.OR.NRP1.EQ.I) GOTO 1                                 02350
      N3=I                                                              02360
      GOTO 2                                                            02370
```

```
    1 CONTINUE
      N3=J
    2 CONTINUE
C
C     SET UP POINTS
C
      X1=AX(N1)
      Y1=AY(N1)
      Z1=AZ(N1)
      X2=AX(N2)
      Y2=AY(N2)
      Z2=AZ(N2)
      X3=AX(N3)
      Y3=AY(N3)
      Z3=AZ(N3)
C
C     COMPUTE COEFFICIENTS
C
      AL1=(((X1-X3)*PCY-(Y1-Y3)*PCX)+(X3*(Y1-Y3)-(X1-X3)*Y3))/
     1             ((X1-X3)*(Y2-Y3)-(X2-X3)*(Y1-Y3))
      AL2=(PCX-AL1*(X2-X3)-X3)/(X1-X3)
      RETURN
      END
      SUBROUTINE AUX(I,J,NLP1,NRP1)
      DIMENSION AX(100),AY(100),AZ(100)
      COMMON N,AX,AY,AZ,PCX,PCY,PCZ,NP1,NP2,NP3
C
C           THIS SUBROUTINE DETERMINES THE NEXT CHORD TO BE COLLAPSED
C     IN THE SEARCH FOR THE BOUNDING SECTION.  IT IS USED INCONJUNCTION
C     WITH EITHER  TRI OR QUAD.
C
      DIMENSION NI(4)
      DIR=(AX(NLP1)-PCX)/ABS(AX(NLP1)-PCX)
      NI(3)=NLP1
      NI(4)=NRP1
      IF(DIR*(AX(I)-PCX)) 4,5,7
    7 CONTINUE
      IF(DIR) 5,4,4
    4 CONTINUE
      NI(1)=I
      NI(2)=J
      GOTO 6
    5 CONTINUE
      NI(1)=J
```

```
      NI(2)=I
    6 CONTINUE
      DM=10000.
C     .
C     FIND THE CHORD FROM AMONG THE FIVE POSSIBLE THAT IS AT MIN
C     DISTANCE FROM THE POINT PC.
C
      DO 1 K=1,3
      N1=NI(1)
      N2=NI(K+1)
      X2=AX(N2)
      Y2=AY(N2)
      DX1=AX(N1)-X2
      DY1=AY(N1)-Y2
      IF(DX1.EQ.0..AND.DY1.EQ.0.) GOTO 2
      AL=((PCX-X2)*DX1+(PCY-Y2)*DY1)/(DX1**2+DY1**2)
      D=(AL*DX1+X2-PCX)**2+(AL*DY1+Y2-PCY)**2
      IF(D.GT.DM) GOTO 2
      KM=K
      DM=D
    2 CONTINUE
    1 CONTINUE
      I=N1
      J=NI(KM+1)
      IF(AX(J).GT.AX(I)) GOTO 3
      I=NI(KM+1)
      J=N1
    3 CONTINUE
      RETURN
      END
      SUBROUTINE QUAD(I,J,NLP1,NRP1,AL1,AL2)
      DIMENSION AX(100),AY(100),AZ(100)
      COMMON N,AX,AY,AZ,PCX,PCY,PCZ,NP1,NP2,NP3
C
C           THIS SUBROUTINE COMPUTES THE INTERSECTION POINT BETWEEN
C     LINE L AND THE PLANE OF THE QUADRILATERAL DEFINED BY THE FOUR
C     POINTS IN THE ARGUMENT LIST.
C
      CALL TRI(I,J,NRP1,J,AL1,AL2)
      IF(AMIN1(AL1,AL2,(1.-AL1-AL2)).LT.0.) GOTO 1
      IF(AMAX1(AL1,AL2,(1.-AL1-AL2)).LE.1.) RETURN
    1 CONTINUE
      CALL TRI(I,NRP1,NLP1,NRP1,AL1,AL2)
      RETURN
```

```
      END
      SUBROUTINE BND(AZMAX,AZMIN,B)
      DIMENSION AX(100),AY(100),AZ(100)
      COMMON N,AX,AY,AZ,PCX,PCY,PCZ,NP1,NP2,NP3,SCALE
C
C           THIS SUBROUTINE EVALUATES THE BOUNDS.
C
      F=(PCZ-AZ(NP2))/(AZ(NP1)-AZ(NP2))
      B=(3.*F-1.)/SCALE+AZMIN
      RETURN
      END
      SUBROUTINE XZROT(I,J)
      DIMENSION AX(100),AY(100),AZ(100)
      COMMON N,AX,AY,AZ,PCX,PCY,PCZ,NP1,NP2,NP3
C
C           THIS SUBROUTINE ROTATES THE SET OF POINTS DEFINED BY THE
C     ARRAYS A IN THE X-Z PLANE   THE ROTATION IS ABOUT POINT PC.  THIS
C     SUBROUTINE IS THE BASIS OF SUBROUTINE CHORD
C
      D1=AX(J)-AX(I)
      D2=AZ(J)-AZ(I)
C
C     CHECK FOR TYPE OF ROTATION
C
      IF(I.NE.NP2.AND.J.NE.NP2) GOTO2
      D3=D2
      D2=D1
      D1=-D3
    2 CONTINUE
C
C     ANGLE PARAMETERS
C
      R=SQRT(D1**2+D2**2)
      CTH=ABS(D1)/R
      STH=D2/R
C
C     ROTATE
C
      DO 1 K=1,NP3
      C1=STH*(AX(K)-PCX)
      C2=STH*(AZ(K)-PCZ)
      AX(K)=CTH*(AX(K)-PCX)+C2+PCX
      AZ(K)=CTH*(AZ(K)-PCZ)-C1+PCZ
    1 CONTINUE
```

```
      RETURN
      END
      SUBROUTINE XYROT(I,J)
      DIMENSION AX(100),AY(100),AZ(100)
      COMMON N,AX,AY,AZ,PCX,PCY,PCZ,NP1,NP2,NP3
C
C           THIS SUBROUTINE ROTATES THE POINTS OF THE ARRAYS A IN THE
C     X-Y PLANE  THIS IS THE CHORD COLLAPSING ROUTINE.
C
      D1=AX(I)-AX(J)
      D2=AY(I)-AY(J)
      R=SQRT(D1**2+D2**2)
      STH=  ABS(D1)/R
      CTH=       D2 /R
      DO 1 K=1,NP3
      C1=STH*(AX(K)-PCX)
      C2=STH*(AY(K)-PCY)
      AX(K)=CTH*(AX(K)-PCX)+C2+PCX
      AY(K)=CTH*(AY(K)-PCY)-C1+PCY
    1 CONTINUE
      RETURN
      END
```

0418 CARDS

Appendix G

This Appendix contains a listing of a FORTRAN IV computer program that solves the four-dimensional moment bounding problem. This program is based on the algorithm described in Chapter III.

```
      DIMENSION AX(102),AY(102),AZ(102),AW(102)
      COMMON N,AX,AY,AZ,AW,PCX,PCY,PCZ,PCW,NP1,NP2,SCALE
      COMMON X(102),Z(102)
C
C           THIS PROGRAM COMPUTES UPPER AND LOWER BOUNDS TO THE FIRST
C     GENERALIZED MOMENT OF A FUNCTION REPRESENTED BY THE ARRAY AZ(K).
C     THE BOUNDS ARE GENERATED BY A FOUR DIMENSIONAL APPLICATION OF
C     AN ISOMORPHISM THEOREM FROM GAME THEORY.  THE APPROXIMATING
C     FUNCTIONS ARE REPRESENTED BY THE ARRAYS AX(K), AY(K), AND AW(K).
C     THE BOUNDING POINT IS GIVEN BY (AM1,AM2,AM3).
C
      N=    50
      H=1.
      DB=-1.
      AM1=.544460858833E-2
      AM2=.830628330745E-4
      AM3=.197265303053E-5
      D=.28334
      AM1=AM1*9.
      AM2=AM2*81.
      AM3=AM3*729.
      D=D*3.
      WRITE(6,13) N,    AM1,AM2,AM3,D
   13 FORMAT(1H1,3HN =,I4,5X,                5HAM1 =,1PE10.3,
     1     5X,5HAM2 =,1PE10.3,5X,5HAM3 =,1PE10.3,5X,
     1                           3HD =,0PF8.5,////)
      DU=D/FLOAT(N-1)
      DO 200 NDB=1,21
      DB=-1.+FLOAT(NDB)
      SIG=10.**(-DB/20.)
      AZMAX=-1.E12
      AZMIN=1.E12
      U=-DU
      DO 1 K=1,N
      U=U+DU
      AX(K)=U**2
      AY(K)=U**4
      AW(K)=U**6
      A1=(H+U)/(SQRT(2.)*SIG)
      A2=(H-U)/(SQRT(2.)*SIG)
      E1=CHEBERF(A1)
      E2=CHEBERF(A2)
      IF(A1.LE.4.) E1=1.-E1
      IF(A2.LE.4.) E2=1.-E2
```

```
      AZ(K)=(E1+E2)/4.
      AZMAX=AMAX1(AZMAX,AZ(K))
      AZMIN=AMIN1(AZMIN,AZ(K))
    1 CONTINUE
      SCALE=1./(AZMAX-AZMIN)
      DO 100 K=1,N
      AX(K)=AX(K)*SCALE
      AY(K)=AY(K)*SCALE
      AW(K)=AW(K)*SCALE
      AZ(K)=(AZ(K)-AZMIN)*SCALE
  100 CONTINUE
C
C     DEFINE LINE L BY ITS END POINTS - INDEXED NP1 AND NP2.
C
      NP1=N+1
      AX(NP1)=AM1*SCALE
      AY(NP1)=AM2*SCALE
      AW(NP1)=AM3*SCALE
      AZ(NP1)=2.
      NP2=N+2.
      AX(NP2)=AM1*SCALE
      AY(NP2)=AM2*SCALE
      AW(NP2)=AM3*SCALE
      AZ(NP2)=-1.
C
C     INITIALIZE ROUTINE FOR LOWER BOUND
C
      ICNT=0
      ICNTL=0
      S=1.
   10 CONTINUE
      DO 110 K=1,NP2
      X(K)=AX(K)
      Z(K)=AZ(K)
  110 CONTINUE
C
C     FIND BOUNDING CHORD FOR THIS INITIAL PROJECTION
C
      ICOL=0
      CALL CHORD(S,I,J,ICOL)
      ICOL=I
C
C     COLLAPSE CHORD IN DEPTH Y
C
```

```
      CALL XYROT(I,J)
C
C     FIND NEW DEPTH Y CHORD
C
      CALL CHORD(S,IY,J,ICOL)
    5 CONTINUE
      ICNT=ICNT+1
C
C      FIND SURFACE
C
      CALL    FACE(I,J,IY,IW,S)
      IF(ICNT.EQ.51) STOP
   14 CONTINUE
C
C     CHECK FOR BCUNDRY
C
      CALL TETRA(I,J,IW,IY,IFLAG)
      IF(IFLAG) 5 ,5 ,51
   51 CCNTINUE
C
C     BOUNDRY FOUND
C
      CALL BND(AZMAX,AZMIN,B)
      IF(S) 8,9,9
    9 CONTINUE
C
C     REINITIALIZE FOR UPPER BOUND
C
      ICNTL=ICNT
      ICNT=0
      S=-1.
      BL=B
      GO TO 10
    8 CONTINUE
      DB=B-BL
      NDBO=NDB-1
      WRITE(6,11) NDBO,B,BL,DB,ICNT,ICNTL
   11 FORMAT(1HO,I10,1P3E16.4,2I10)
  200 CONTINUE
      STOP
      END
      SUBROUTINE CHORD(S,I,J,ICOL)
      DIMENSION AX(102),AY(102),AZ(102),AW(102)
      DIMENSION LL(100),LR(100)
```

```
      COMMON N,AX,AY,AZ,AW,PCX,PCY,PCZ,PCW,NP1,NP2,SCALE
      COMMCN X(102),Z(102)
C
C            THIS SUBROUTINE FINDS THE BOUNDING CHORD CH. FOR S POSITIVE,
C     IT FIND THE LOWER BOUND, WHILE FOR S NEG. IT FIND THE UPPER BOUND.
C
      LCL=0
      LCR=0
      IF(S) 21,22,22
   22 CONTINUE
      PL=100.
      PR=100.
      I=NP1
      IF(ICOL.NE.0) GOTO 10
      J=NP1
      GOTO 10
   21 CONTINUE
      PL=-1000.
      PR=-1000.
      I=NP2
      IF(ICOL.NE.0) GOTO 10
      J=NP2
   10 CONTINUE
      XINT=X(NP2)
      IF(ICOL.EQ.0) GOTO 50
      PR= Z(J)
      XINT= X(J)
   50 CONTINUE
      DO 2 K=1,N
C
C     SPLIT INTO LEFT AND RIGHT OF LINE L.- FIRST PASS
C
      IF(K.EQ.ICOL) GOTO 2
      IF( X(K)-XINT) 3,4,4
    3 CONTINUE
      LCL=LCL+1
      LL(LCL)=K
C
C     BELOW (ABOVE) LINE L
C
      IF(S*( Z(K)-PL)) 5,7,6
    5 CONTINUE
      I=K
      PL= Z(K)
```

```
      GOTO 6
    7 CONTINUE
      IF( X(K)- X(I)) 6,6,5
    4 CONTINUE
      LCR=LCR+1
      IF(ICOL.NE.0) GOTO6
      LR(LCR)=K
      IF(S*( Z(K)-PR)) 8,9,6
    8 CONTINUE
      J=K
      PR= Z(K)
      GOTO 6
    9 CONTINUE
      IF( X(K)- X(J)) 8,6,6
    6 CONTINUE
    2 CONTINUE
      IF(LCL.NE.0.AND.LCR.NE.0) GOTO11
      WRITE(6,20)
   20 FORMAT(1H0,30HLINE L DOES NOT INTERSECT HULL)
      STOP
   11 CONTINUE
C
C     FIND POINT PC
C
C
C     LEVEL CHORD CH
C
      CALL XZROT(I,J,3)
C
C     INITIALIZE AND BEGIN CHORD CHECK SEQUENCE
C
      PT=PCZ
      KL=I
      KR=J
      PTL=PT
      PTR=PT
      DO 14 M=1,LCL
      K=LL(M)
      IF(S*( Z(K)-PTL)) 18,13,13
   18 CONTINUE
      PTL= Z(K)
      KL=K
   13 CONTINUE
   14 CONTINUE
```

```
16 CONTINUE                                                        02060
   IF(ICOL.GT.0) GOTO 53
   DO 15 M=1,LCR
   K=LR(M)
   IF(S*( Z(K)-PTR)) 19,12,12
19 CONTINUE                                                        02080
   PTR= Z(K)                                                       02090
   KR=K                                                            02100
12 CONTINUE                                                        02120
15 CONTINUE                                                        02000
53 CONTINUE
   IF(KL.EQ.I.AND.KR.EQ.J) GOTO 17                                 02130
   I=KL                                                            02140
   J=KR                                                            02150
   PL=PTL                                                          02160
   PR=PTR                                                          02170
   GOTO 11                                                         02180
17 CONTINUE                                                        02190
   RETURN                                                          02200
   END                                                             02210
   SUBROUTINE BND(AZMAX,AZMIN,B)                                   03150
   DIMENSION AX(102),AY(102),AZ(102),AW(102)                       03160
   COMMON N,AX,AY,AZ,AW,PCX,PCY,PCZ,PCW,NP1,NP2,SCALE              03170
C                                                                  03180
C        THIS SUBROUTINE EVALUATES THE BOUNDS.                     03190
C                                                                  03200
   F=(PCZ-AZ(NP2))/(AZ(NP1)-AZ(NP2))                               03210
   B=(3.*F-1.)/SCALE+AZMIN
   RETURN                                                          03230
   END                                                             03240
   SUBROUTINE XZROT(I,J,IND)                                       03250
   DIMENSION AX(102),AY(102),AZ(102),AW(102)                       03260
   COMMON N,AX,AY,AZ,AW,PCX,PCY,PCZ,PCW,NP1,NP2,SCALE              03270
   COMMON X(102),Z(102)
C                                                                  03280
C        THIS SUBROUTINE ROTATES THE SET OF POINTS DEFINED BY THE  03290
C  ARRAYS A IN THE X-Z PLANE   THE ROTATION IS ABOUT POINT PC.  THIS 03300
C  SUBROUTINE IS THE BASIS OF SUBROUTINE CHORD                     03310
C                                                                  03320
   D1=X(J)-X(I)
   D2=Z(J)-Z(I)
   CALL PC(I,J,1)
C
C  ANGLE PARAMETERS
```

```
C
      R=SQRT(D1**2+D2**2)
      CTH=ABS(D1)/R
      STH=D2/R
C
C     ROTATE
C
      DO 3 K=1,NP2
      C1=STH*( X(K)-PCX)
      C2=STH*( Z(K)-PCZ)
       X(K)=CTH*( X(K)-PCX)+C2+PCX
       Z(K)=CTH*( Z(K)-PCZ)-C1+PCZ
    3 CONTINUE
      RETURN
      END
      SUBROUTINE XYROT(I,J)
      DIMENSION AX(102),AY(102),AZ(102),AW(102)
      COMMON N,AX,AY,AZ,AW,PCX,PCY,PCZ,PCW,NP1,NP2,SCALE
      COMMON X(102),Z(102),Y(102)
C
C           THIS SUBROUTINE ROTATES THE POINTS OF THE ARRAYS A IN THE
C     X-Y PLANE.  THIS IS THE DEPTH Y CHORD COLLAPSING ROUTINE.
C     CHORDS ARE ALWAYS COLLAPSED SUCH THAT THEY ARE TO THE RIGHT
C     OF THE LINE L IN THE X-Z PROJECTION.
C
      D1= X(I)- X(J)
      D2=AY(I)-AY(J)
      S=D2/D1
      X0=( X(J)*S**2+(PCY-AY(J))*S+PCX)/(1.+S**2)
      Y0=S*(X0- X(J))+AY(J)
      S2=(Y0-PCY)/ABS(Y0-PCY)
      IF(S.GE.0.) S1=-S2
      IF(S.LT.0.) S1=S2
      R=SQRT(D1**2+D2**2)
      STH=S2*ABS(D1)/R
      CTH=S1*ABS(D2)/R
      DO 1 K=1,NP2
      C1=STH*(X(K)-PCX)
      C2=STH*(AY(K)-PCY)
       X(K)=CTH*( X(K)-PCX)+C2+PCX
      Y(K)=CTH*(AY(K)-PCY)-C1+PCY
    1 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE TETRA(I,J,IW,IY,IFLAG)
      DIMENSION AX(102),AY(102),AZ(102),AW(102)
      DIMENSION TET(3)
      COMMON N,AX,AY,AZ,AW,PCX,PCY,PCZ,PCW,NP1,NP2,SCALE
C
C           THIS SUBROUTINE DETERMINES IF THE FOUR POINT SURFACE
C     FEATURE DEFINED BY THE POINTS (I,J,IW,IY) IS THE FEATURE THAT
C     DEFINES THE BOUND.  SUCCESS IS INDICATED BY IFLAG BEING POSITIVE.
C
C
C     SET-UP COEFFICIENT COMPUTATION
C
      A0=AX(NP2)-AX(IW)
      A1=AX(I)-AX(IW)
      A2=AX(J)-AX(IW)
      A3=AX(IY)-AX(IW)
      B0=AY(NP2)-AY(IW)
      B1=AY(I)-AY(IW)
      B2=AY(J)-AY(IW)
      B3=AY(IY)-AY(IW)
      D0=AZ(NP2)-AZ(IW)
      D1=AZ(I)-AZ(IW)
      D2=AZ(J)-AZ(IW)
      D3=AZ(IY)-AZ(IW)
      C0=AW(NP2)-AW(IW)
      C1=AW(I)-AW(IW)
      C2=AW(J)-AW(IW)
      C3=AW(IY)-AW(IW)
C
C     COMPUTE COEFFICIENTS
C
      AL3=((A1*C0-A0*C1)*(A1*B2-A2*B1)-(A1*B0-A0*B1)*(A1*C2-A2*C1))/
     1       ((A1*C3-A3*C1)*(A1*B2-A2*B1)-(A1*B3-A3*B1)*(A1*C2-A2*C1))
      AL2=((A1*B0-A0*B1)-AL3*(A1*B3-A3*B1))/(A1*B2-A2*B1)
      AL1=(A0-AL2*A2-AL3*A3)/A1
      AL4=1.-AL1-AL2-AL3
C
C     CHECK COEFFICIENTS
C
      IF(AMIN1(AL1,AL2,AL3,(1.-AL1-AL2-AL3)).LT.0.) GOTO 2
      IF(AMAX1(AL1,AL2,AL3,(1.-AL1-AL2-AL3)).GT.1.) GOTO 2
      PCZ=AL1*D1+AL2*D2+AL3*D3+AZ(IW)
    3 CONTINUE
      IFLAG=1
```

```
      RETURN                                                            TET  620
    2 CONTINUE                                                          TET  630
      IFLAG=-1                                                          TET  640
C
C      DETERMINE POINT TO BE CROPPED
C
      TET(1)=AL1
      TET(2)=AL2
      TET(3)=AL3
      TH=-1000000.
      DO 4 INDEX=1,3
      IF(TET(INDEX).GT.0.) GOTO 5
      IF(TET(INDEX).LT.TH) GOTO 5
      ITH=INDEX
      TH=TET(INDEX)
    5 CONTINUE
    4 CONTINUE
      GOTO (6,7,8),ITH
    6 CONTINUE
      I=IW
      RETURN
    7 CONTINUE
      J=IW
      RETURN
    8 CONTINUE
      IY=IW
      RETURN                                                            TET  650
      END                                                               TET  730
      SUBROUTINE PC(I,J,IND)
      DIMENSION AX(102),AY(102),AZ(102),AW(102)
      COMMON N,AX,AY,AZ,AW,PCX,PCY,PCZ,PCW,NP1,NP2,SCALE
      COMMON X(102),Z(102)
      DCZ= Z(I)- Z(J)
      DLX= X(NP1)- X(NP2)
      DLZ= Z(NP1)- Z(NP2)
      DLW=AW(NP1)-AW(NP2)
      DLY=AY(NP1)-AY(NP2)
      IF(DCZ) 5,6,5
    6 CONTINUE
      PCZ=Z(I)
      GOTO 7
    5 CONTINUE
      DCX= X(I)- X(J)
      F1=DLX*DCZ
```

```
      F2=DLZ*DCX
      PCZ=( Z(J)*F2+( Z(NP1)* X(NP2)- X(NP1)* Z(NP2)- X(J)*DLZ)*
     1     DCZ)/(F2-F1)
    7 CONTINUE
      PCX=(PCZ- Z(NP2))*DLX/DLZ+ X(NP2)
      PCY=(PCZ- Z(NP2))*DLY/DLZ +AY(NP2)
      PCW=(PCZ- Z(NP2))*DLW/DLZ+AW(NP2)
      RETURN
      END
      SUBROUTINE     FACE(I,J,K,L,S)
      DIMENSION AX(102),AY(102),AZ(102),AW(102)
      COMMON N,AX,AY,AZ,AW,PCX,PCY,PCZ,PCW,NP1,NP2,SCALE
      DIMENSION NK(100)
      ZTT=1000.*S
      KN1=0
C
C      INITIALIZE ROUTINE
C
      X1=AX(I)
      Y1=AY(I)
      Z1=AZ(I)
      W1=AW(I)
      X2=AX(J)-X1
      X3=AX(K)-X1
      Y2=AY(J)-Y1
      Y3=AY(K)-Y1
      W2=AW(J)-W1
      W3=AW(K)-W1
      Z2=AZ(J)-Z1
      Z3=AZ(K)-Z1
      CX=Y2*W3-Y3*W2
      CY=X3*W2-X2*W3
      CW=X2*Y3-X3*Y2
      F1=CY/CX
      F2=CW/CX
C
C      CLASSIFY POINTS IN THE SET
C
      DO 1000 M1=1,NP1
      M=NP2-M1
      IF(M.EQ.I.OR.M.EQ.J.OR.M.EQ.K) GOTO 1001
      Y0=AY(M)-Y1
      W0=AW(M)-W1
      D=AX(M)-X1+Y0*F1+W0*F2
```

BEST AVAILABLE COPY

```
      IF(M1.GT.1) GOTO 1002
C
C      SET-UP TEST
C
      DTEST=D/ABS(D)
      DTH=1000.*DTEST
      GOTO 1001
 1002 CONTINUE
C
C     TEST POINTS
C
      IF(D*DTEST) 1001,1001,1009
 1009 CONTINUE
C
C      DESIGNATE POINTS OF INTEREST
C
      KN1=KN1+1
      NK(KN1)=M
      IF(S*(AZ(M)-ZTT)) 1014,1018,1001
 1018 CONTINUE
      IF(ABS(D).GT.ABS(DTH)) GOTO 1001
 1014 CONTINUE
C
C      DETERMINE LOWEST IN Z-SENSE
C
      DTH=D
      KZ=M
      ZTT=AZ(M)
 1001 CONTINUE
 1000 CONTINUE
      IF(KN1.GT.0) GOTO 1016
 1023 CONTINUE
      WRITE(6,1017)
 1017 FORMAT(1H0,15HSURFACE FAILURE)
      STOP
 1016 CONTINUE
      KNO=KN1+2
      NCYC=0
      NPTS=0
 1010 CONTINUE
      ZT=ZTT
      KN=KN1
      NCYC=NCYC+1
      IF(KNO.EQ.NCYC) GOTO 1023
```

```
      NPTS=NPTS+KN1
C
C      SET-UP SURFACE CHECK
C
      X4=AX(KZ)-X1
      Y4=AY(KZ)-Y1
      Z4=AZ(KZ)-Z1
      W4=AW(KZ)-W1
      FX=Y4*(Z2*W3-Z3*W2)+Y2*(Z3*W4-Z4*W3)+Y3*(Z4*W2-Z2*W4)
      FY=X4*(Z2*W3-Z3*W2)+X2*(Z3*W4-Z4*W3)+X3*(Z4*W2-Z2*W4)
      FZ=X4*(Y2*W3-Y3*W2)+X2*(Y3*W4-Y4*W3)+X3*(Y4*W2-Y2*W4)
      FW=X4*(Y2*Z3-Y3*Z2)+X2*(Y3*Z4-Y4*Z3)+X3*(Y4*Z2-Y2*Z4)
      FO=-X1*FX+Y1*FY-Z1*FZ+W1*FW
      FO=-FO/FZ
      FX=-FX/FZ
      FY=FY/FZ
      FW=FW/FZ
      KN1=0
      ZTT=1000.*S
      KZO=KZ
      DO 1011 MI=1,KN
C
C      CHECK FOR VALID SURFACE
C
      M=NK(MI)
      IF(M.EQ.KZO) GOTO 1012
      Z=FO+FX*AX(M)+FY*AY(M)+FW*AW(M)
      IF(S*(AZ(M)-Z)) 1013,1012,1012
 1013 CONTINUE
      KN1=KN1+1
      NK(KN1)=M
      IF(S*(AZ(M)-ZTT))1015,1012,1012
 1015 CONTINUE
      KZ=M
      ZTT=AZ(M)
 1012 CONTINUE
 1011 CONTINUE
      IF(KN1.NE.0) GOTO 1010
      L=KZ
      RETURN
      END
```

0525 CARDS

REFERENCES

[1] Lucky, R. W., J. Salz, and E. J. Weldon, Jr., Principles of Data Communication, McGraw-Hill, New York, 1968.

[2] Ho, E. Y., and Y. S. Yeh, "A New Approach for Evaluating the Error Probability in the Presence of Intersymbol Interference and Additive Gaussian Noise," Bell Syst. Tech. J., Vol. 49, pp. 2249-2266, Nov. 1970.

[3] Shimbo, O. and M. I. Celebiler, "The Probability of Error Due to Intersymbol Interference and Gaussian Noise in Digital Communication Systems," IEEE Trans. Comm. Tech., Vol. COM-19, pp. 113-119, April 1971.

[4] Prabhu, V. K., "Some Considerations of Error Bounds in Digital Systems," Bell Syst. Tech. J., Vol. 50, pp. 3127-3151, Dec. 1971.

[5] Benedetto, S., G. DeVincentiis, and A. Luvison, "Error Probability in the Presence of Intersymbol Interference and Additive Noise for Multilevel Digital Systems," IEEE Trans. Comm. Tech., Vol. COM-21, pp. 181-188, March 1973.

[6] Saltzberg, B. R., "Intersymbol Interference Error Bounds with Application to Ideal Bandlimited Signaling," IEEE Trans. Inform. Theory, Vol. IT-14, pp. 563-568, July 1968.

[7] Lugannani, R., "Intersymbol Interference and Probability of Error in Digital Systems," IEEE Trans. Inform. Theory, Vol. IT-15, pp. 682-688, Nov. 1969.

[8] Glave, F. E., "An Upper Bound on the Probability of Error Due to Intersymbol Interference for Correlated Digital Signals," IEEE Trans. Inform. Theory, Vol. IT-18, pp. 356-363, May 1972.

[9] Matthews, J. W., "Sharp Error Bounds for Intersymbol Interference," IEEE Trans. Inform. Theory, Vol. IT-19, pp. 440-~~448,~~ July 1973.

[10] McLane, P. J., "Lower Bounds for Finite Intersymbol Interference Error Rates," IEEE Trans. Comm. Tech., Vol. COM-22, pp. 853-857, June 1974.

[11] Yao, K. and R. M. Tobin, "Moment Space Upper and Lower Error Bounds for Digital Systems with Intersymbol Interference," IEEE Trans. Inform. Theory, Vol. IT-22, pp. 65-74, Jan. 1976.

[12] Yan, T. Y., Moment Space Error Bounds for Digital Communication Systems with Intersymbol Interference Based on N^{th} Moment, M. S. in Engineering, Systems Science Department, University of California, Los Angeles, Jan. 1976.

[13] Yao, K., "Quadratic-Exponential Moment Error Bounds for Digital Communication Systems," Conf. Rec. of the Tenth Annual Asilomar Conf. on Circuits, Systems, and Computers, Pacific Groves, Calif., Nov. 1976, pp. 99-103.

[14] Yao, K. and R. M. Tobin, "Multi-Dimensional Moment Space Error Bounds for Digital Systems with Multiple Interferences," Abstract of Papers of the 1976 Int. Symposium on Information Theory, Ronneby, Sweden, June, 1976, p. 69.

[15] Yao, K. and E. Biglieri, "Moment Inequalities for Error Probabilities in Digital Communication Systems," Proc. of the Nat. Telecommunication Conf., Los Angeles, Calif., Dec. 1977.

[16] Tobin, R. M., and K. Yao, "Upper and Lower Error Bounds for Coherent Phase-Shift-Keyed (CPSK) Systems with Cochannel Interference," IEEE Trans. Comm. Tech., Vol. COM-25, pp. 281-287, Feb. 1977.

[17] Yao, K., "Error Probability of Asynchronous Spread Spectrum Multiple Access Communication Systems," IEEE Trans. Comm. Tech., Vol. COM-25, pp. 803-809, Aug. 1977.

[18] Yao, K., "Error Proability of Spread Spectrum Multiple Access Communication Systems," Proc. of the 1976 Conf. on Information Sciences and Systems, John Hopkins Univ., Baltimore, Maryland, March 1976, pp. 67-72.

[19] Yao, K., Performance Bounds on Spread Spectrum Multiple Access Communication Systems," Proc. of the International Telemetry Conf., Los Angeles, Calif., Sept. 1976, pp. 317-325.

[20] Dresher, M., S. Karlin, L. S. Shapley, "Polynomial Games," Contributions to the Theory of Games, Annals of Mathematics Studies, No. 24, 1950, pp. 161-180.

[21] Dresher, M., "Moment Spaces and Inequalities," Duke Math. J., Vol. 20, pp. 261-271, June 1953.

[22] Rockafellar, R. T., Convex Analysis, Princeton University Press, Princeton, New Jersey, 1970.

[23] Balakrishnan, A. V., Applied Functional Analysis, Springer-Verlag, New York, 1976.

[24] Eisenhart, L. P., A Treatise on the Differential Geometry of Curves and Surfaces, Dover, New York, 1960.

[25] Wylie, C. R. Jr., Advanced Engineering Mathematics, 2nd Edition, McGraw-Hill, New York, 1960.

[26] Appel, A., and P. M. Will, "Determining the Three-Dimensional Convex Hull of a Polyhedron," IBM J. of Research and Development, pp. 590-601, Nov. 1976.

[27] Preparata, F. P., and S. J. Hong, "Convex Hulls of Finite Planar and Spatial Sets of Points," Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, Illinois, Report R-682 (UILU-ENG 75-2217), April, 1975.

[28] Wilhelmsen, D. R., "A Nearest Point Algorithm for Convex Polyhedral Cones and Applications to Positive Linear Approximation," Mathematics of Computation, Vol. 30, No. 133, pp. 48-57, January 1976.

[29] Balinski, M. L., "An Algorithm for Finding all Verticies of Convex Polyhedral Sets," J. Soc. Indust. Appl. Math., Vol. 9, No. 1, pp. 72-88, March 1961.

[30] Kendall, M. G., A Course in the Geometry of n Dimensions, Charles Griffin and Co. Ltd., London, 1961.

[31] Grünbaum, B., Convex Polytopes, Interscience Publishers, New York, 1967.

[32] Busemann, H., Convex Surfaces, Interscience Publishers, New York, 1967.

[33] Lyusternik, L. A., Convex Figures and Polyhedra, D. C. Heath and Co., Boston, 1966.

[34] King, M. A., "Three Dimensional Geometric Moment Space Bounds with Application to Problems in Communications Theory," Systems Science Department, University of California, Los Angeles, UCLA-ENG-7805, February 1978.